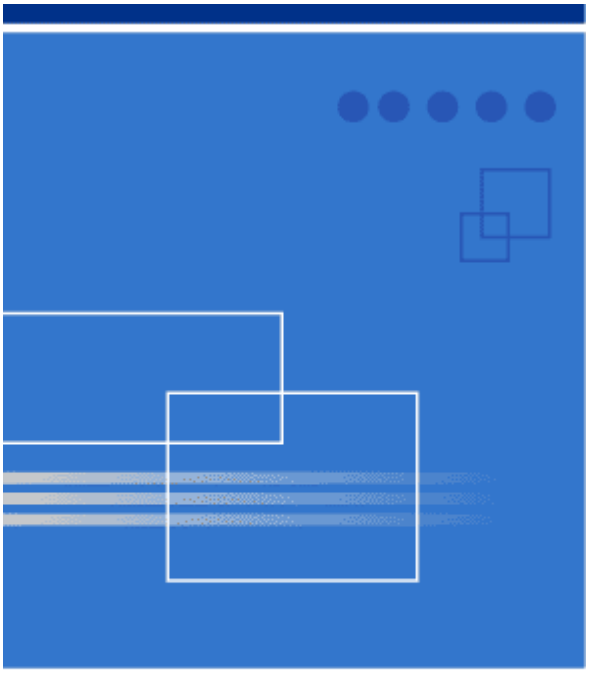


CoMET®

user guide

VaST SYSTEMS TECHNOLOGY
USA • JAPAN • EUROPE • AUSTRALIA

VaST Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

CoMET

Version 5.14

User Guide

VaST SYSTEMS TECHNOLOGY

Confidential and Proprietary.

May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Document Version No: 1.0

© Copyright 1997 – 2006 VaST Systems Technology Corp. All rights reserved

The copyright owner of CoMET hereby disclaims all warranties relating to this software, whether express or implied, including without limitation any implied warranties of merchantability or fitness for a particular purpose. The copyright owner will not be liable for any special, incidental, consequential, indirect or similar damages due to loss of data or any other reasons, even if the possibility of such damages has been advised. The person using the software shall bear all the risks as to the consequences of using this software.

Contents

Contents ..... **i**

Introduction **1**

About This Manual 2

References 2

What's New in CoMET 3

New in CoMET 5.14 3

Visual C++ Version 8 and Version 8 Express Supported 3

Implemented Support for –mno–cygwin Option with Gcc Projects 3

New Buttons Added to Toolbar 3

New in CoMET 5.13 3

Verilog Support 3

Multiple Versions of Models Support 3

New Parameters Added to the Default Parameter Views 3

Enabled Sorting in Versions Dialog Box 3

New in CoMET 5.12 4

VaST VPM Transformer Toolset 4

New in CoMET 5.11 4

Visual C++ Version 8 Supported 4

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly
authorized in a written agreement with VaST.

Installing CoMET 5
CoMET Components 5
Installing CoMET on Windows XP or Windows 2000 5
Setting up Environment Variables 6
Environment Variables 6
Changing Environment Variables in Windows XP or Windows 2000 7
Installing CoMET on a Network 8
Preparation 8
CoMET Installation 8
Copying a CoMET Installation 8
On each Workstation 9
Set DTD Locations 9
Set XML Reports Locations 10
Uninstalling CoMET 10
Getting Started 11
Starting CoMET in Windows XP/2000 11
The CoMET System Engineering Environment 12
Menu Bar 12
Tool Bar 13
Status Bar 13
Workspace Window 13
Target Window 13
Output Windows 13
Document Windows 14
Context Menu 14
Modules Window 14
Multiple Versions of Models within CoMET 14
Maintaining Multiple Versions of a Module CoMET 16
Toolbar Buttons 17
“New File” Button 17
Workspace Button 18
Log Button 18
Target Button 19

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly
authorized in a written agreement with VaST.

Doing the Tutorials 19
Windows XP Theme Display Issue 20
Arranging Windows 22
Document windows 22
Tool windows 22
Docking Tool Windows 22
Auto Hide Button 25
Configuring the CoMET System Engineering Environment 26
General Configuration Settings 26
DTD Locations 27
Default XML Views 27
Runtime Options: Enable runtime dependency checks 28
Runtime Options: Auto-convert Gcc projects to No Cygwin Dll 28
Editor Options Configuration Settings 29
Tool Locations Configuration Settings 31
User Locations Configuration Settings 32
XML Tables Configuration Settings 33
Parameter Table Editing 34
Connection Table editing 36
Modules Configuration Settings 38
Environment Configuration Settings 40
Displaying CoMET and Tools / Models Versions 41
Getting Assistance while You Work 42
CoMET Online Links 42
CoMET Web Page 42
CoMET Support Services 42

Managing Workspaces and Projects 43

Creating a Workspace 43
Adding a New Project 45
Defining a Fabric Module 47
Module Class 47
Module Type 48
Version 49

Project Configuration....49
Adding an Existing Project....50
Removing a Project from the Workspace....51
Viewing the Properties of a Project....52
Setting the Active Project....53
Configuring a Project....54
Adding a Source File to a Project....55
Viewing Project Component Properties....56
Properties of Special Components....56
Viewing the File Path Property....56
Disabled Properties....56
Saving Workspace Files....57
Exporting a Virtual System Prototype Workspace for METeor....57
Exporting a Workspace....57
Choosing Export Options....58
Files Exported for Specified Projects....58
Ensuring the Fabric project Prototype Configuration (.pcx) file is exported....58
Exporting Additional Files and Folders....59
Location of extra files and folders in Workspace....60
Creating the Exported Workspace .cab File....60
Moving a workspace from one directory to another....61

Developing a Fabric Module Definition....63
Opening the Fabric Module Definition file....63
Editing the Fabric Module Definition file....66
Prerequisite Elements....66
Choosing Names....67
Description attributes....67
Arrays....67
Declaring an Array....67
Array Notation Declaration....67
The Number Of Attribute....68
Referring to Members of an Element Array....69
Using Array Notation in an Instance Port Name....69

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly
authorized in a written agreement with VaST.

Using Array Notation in a Port Connection Port Or Net Name 70
Using Array Notation in a Port or Net View 70
Options for Referring to Array Members 71
Fmx Update - Update Fabric 72
Child Elements 72
Creating a Child Element 72
The Element Dialog 73
FMX Hierarchy 74
Hierarchical Parameter Type Naming 76
Fabric Module 77
Parameters 78
Parameter 78
Sub Parameters 79
Ports 80
Port 80
Nets 81
Net 81
Tasks 82
Tasks 82
Task 82
Port Or Net Views 83
Port Or Net View 83
Module Instances 84
Module Instance 84
Adding a Module Instance using the Context Menu 86
Adding a Module Instance using Drag and Drop from the Modules window 87
Adding a Module Instance using Copy and Paste 88
Parameter Overrides 89
Parameter Override 89
Parameter Fmx Access, Pcx Access and Parameter Overrides 90
The Pcx Access Field is Always Editable in a Parameter Override 91
Changing the Override Value 92
Sub Parameter Overrides 93
Instance Ports 94

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly
authorized in a written agreement with VaST.

Instance Port....94
Port Connection....95
Interfaces....96
Interface....96

Views and Reports....97
Using Table Views....97
Changing Column Order....97
Parameter Views....98
Connection Views....99
Viewing Module Definition and Prototype Configuration Reports....100
Viewing a Fabric Module Definition Report - Fmx Report....100
Fmx Report Errors and Warnings....104
Viewing a Prototype Configuration Report - Pcx Report....105
Viewing All Connections in a Virtual System Prototype....106

Configuring Build and Runtime Options....109
Setting the Build Configuration....109
Build Configuration Types....110
Common Build Configuration Properties....110
CIF Project Build Configuration Properties....111
VBus Standalone Build Configuration Parameters....112
Tab files are no longer required....112
VBus CIF Build Configuration Properties....113
Target Project Build Configuration Properties....114
Setting the Active Build Configuration....114
Setting the Runtime Options....117
Adding Runtime Option Properties....117
Viewing and Setting Runtime Options Properties....118
Fabric Project Runtime Options....119
VBus Module Runtime Options Properties....120
Supported simulation engines....120
Modifying Parameter Values in the Prototype Configuration File....121
Pcx Update - Update Platform....122
Excluding Build-Time Parameters from Prototype Configuration Files....122

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly
authorized in a written agreement with VaST.

Setting the Target Executable 124
Target Image Properties 125
Restrictions on the Target Image Comparator SREC format 127

Using the Source Editor 129
Highlighting Keywords 131
Editor features 131
Find in Files 132
Locating Symbols with the Browse Database 134
Goto Definition 134
Browse Symbol 135
Special Editing Keys 136

Building the Workspace and Projects 137

Simulating and Debugging a Model 139
Choosing a Debugger 139
Hardware Model Debugging 139
GDB Internal Debugger 140
CDB Internal Debugger 140
Target Code Debugging 140
Executing in Simulation Mode and Debug Mode 141
Interrupting the Execution Flow 142
Setting Breakpoints in Hardware Model Debugging 142
Managing Breakpoints 143
Continuing Execution 143
Continue Debugging 143
Step Into 144
Step Over 144
Step Out 144
Using the Target Debugger Break Button 144
Validating VPM behavior 145
Validate 145
Validate All 145
Obtaining a Value Change Dump (VCD) 146
Building and Simulating in Batch Mode 147

Specifying Batch Mode Options using the Batch Mode Dialog....147
Build options....148
Prototype Options....148
Target options....148
Target Time Limits....149
Saving the batch....149
Exporting a CoMET Batch File....149
Importing a CoMET batch file....149
Running the batch....149
Command line batch mode....150
Batch Mode Options....150
Non-Batch-Mode Options....150
CoMET Batch File....150
Command line batch mode using a CoMET 4 batch file....151
Running External Tools during Simulation....152
Target Debugging Considerations....154

Hot Keys....155
Hot Keys - All Windows....155
Hot Keys - All Windows - Files....155
Hot Keys - All Windows - Editing....156
Hot Keys - Viewing CoMET Windows....156
Hot Keys - Workspace Window....156
Hot Keys - Document Window....156
Hot Keys - Document Window - Source Code Navigation and Bookmarks....157
Hot Keys - Building Projects....157
Hot Keys - Simulating and Debugging....157

CoMET 4 and SystemC Compatibility....159
Converting CoMET 4 projects....159
Using the Matlab Generator....159
Running a VBus Peripheral Model on CIF....160
Creating a VBus Peripheral Model Project....160
Parameters....163
Specifying the VBus device type....163

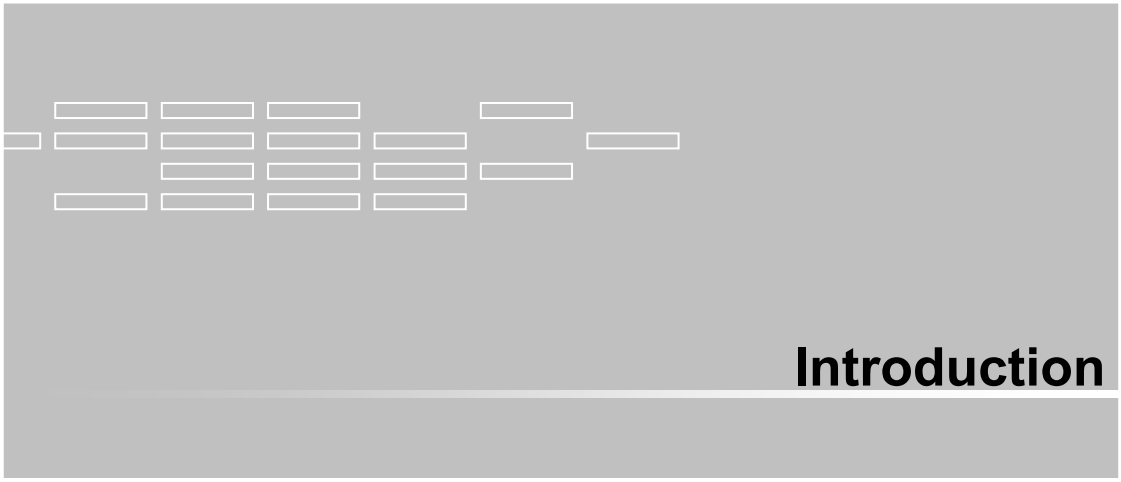
VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly
authorized in a written agreement with VaST.

Simulating the VBus Peripheral Model 163
Simulating a VBus Peripheral Model with FinSim 163
Creating SystemC Projects 165
Creating a Standalone SystemC Project 165
Creating a CIF SystemC Project 165
Choosing the SystemC Libraries for VC++6, VC++7, VC++8 and GCC 166

ModelSim HDL Co-Simulation Support 169
Introduction to ModelSim HDL Co-Simulation Support 169
VHDL Co-Simulation Support 169
Features and requirements 169
Mappings between VHDL and Fabric Port 170
Running a VHDL Model on CIF 171
Tool Locations Configuration Settings for VHDL Model 171
Creating a VHDL Model Project 172
Parameters 175
Selecting a Fabric Port Type for a VDHL Port 176
Setting the Runtime Options 177
Verilog Co-Simulation Support 180
Features and requirements 180
Mappings between Verilog and Fabric Port 180
Running a Verilog Model on CIF 183
Tool Locations Configuration Settings for Verilog Model 183
Creating a Verilog Model Project 184
Selecting a Fabric Port Type for a Verilog Port 188
Setting the Runtime Options 189

Glossary 193

Index 197



Introduction

Welcome to CoMET (Co-Design Methodology and Engineering Toolset), an integrated development environment for building VaST Virtual System Prototypes.

From within CoMET you can model, verify and debug complete hardware-software systems in systems-on-chip (SOC), application specific integrated circuits (ASIC), and printed circuit board (PCB) designs.

A Virtual System Prototype is a virtual (computer-based) model of a hardware/software system that you can use to develop system architecture. With a VSP you can simulate and analyze system performance before you develop and manufacture the actual hardware. You can incorporate incremental detailed subsystem design as it becomes available. A VSP also provides you with a precise format for communicating specifications to the software and hardware design groups.

With a CoMET Virtual System Prototype you can develop and debug hardware and software concurrently, instead of sequentially. A VSP provides complete system verification across all components, resource usage data, and performance quantification at every level of design.

CoMET supports the specification and timing-accurate modeling of electronic hardware, buses and networks. Using a VSP you can develop and run software applications, operating systems, middleware and drivers, and test telecommunications protocols and DSP algorithms, prior to hardware availability.

CoMET allows you to quantitatively experiment early in the engineering process with hardware/software partitioning, choice of processors and co-processors (and configuration), and operating system options. You can use CoMET to evaluate trade-offs in design such as cache sizing, clock speed, and processor type – throughout the design process – without sacrificing timing or accuracy. You can test more of the software before realization, and experiment with various processor combinations, performing "what if" analyses to determine optimal designs.

With CoMET you can move the boundary between hardware and software, and view and control the interaction between hardware and software interactively – even in highly

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly
authorized in a written agreement with VaST.

concurrent designs having several different processors and considerable application-specific hardware.

CoMET is ideally suited for complex system-on-chip (SOC) designs, including multi-processor systems. It simulates complete system-level behavior, function, and timing. CoMET is also ideal for larger system designs such as ECUs in automotive, or even full distributed systems of ECUs.

CoMET supports incremental integration of components as they are completed and tested throughout the engineering process.

CoMET executes application software on retarget-able Virtual Processor Models™ (VPM™) at speeds in excess of 50,000,000 instructions/second while preserving timing accuracy across the hardware/software interface.

About This Manual

This document discusses how to install and use VaST Systems Technology's CoMET System Engineering Environment tool, and how to build, simulate and debug a Virtual System Prototype.

There are several topics on system design beyond the scope of this document. The definition of system architecture and how to make a fast, efficient model is not included in this document.

Also beyond the scope of this document is a description of VaST's Communication and Infrastructure Fabric, Virtual Processor Models, Virtual Architecture Platforms and the following add-on modules or features:

- The specific virtual processor models (VPMs) available from VaST Systems Technology.
- Metrix tracing, profiling and dynamic debugging methodology.
- Integrating Physical models (such as Matlab).

References

You should use this manual in conjunction with the following:

- CoMET v5.14 Tutorial
- CoMET Modeling Guide (CIF)
  We recommend reading the Modeling Guide before reading this document.
- Metrix User Guide
- Metrix Configuration File User Guide
- CoMET Modeling Guide v1.0 - if working with VBus projects
- CoMET and METeor Third Party Tool Support - integration with debuggers and other third party tools

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

What's New in CoMET

New in CoMET 5.14

Visual C++ Version 8 and Version 8 Express Supported

CoMET now supports Visual C++ Version 8 and Version 8 Express. See *Tool Locations Configuration Settings, page 31*.

Implemented Support for –mno–cygwin Option with Gcc Projects

Implemented support for the –mno–cygwin option with Gcc projects. See *General Configuration Settings, page 26*.

New Buttons Added to Toolbar

The following buttons have been added to the Toolbar: New File, Workspace, Log and Target. See *Toolbar Buttons, page 17*.

New in CoMET 5.13

Verilog Support

The ModelSim HDL Co-Simulation now supports Verilog models. See *Verilog Co-Simulation Support, page 180*.

Multiple Versions of Models Support

CoMET supports multiple versions of a model to be located in the modules (%VAST5_HOME%\Modules) directory. See *Multiple Versions of Models within CoMET, page 14*.

New Parameters Added to the Default Parameter Views

- Added MemoryWidth, Tunneling, NoOfMemoryPages and PageSize to the default Memory Parameter view. See *Parameter Views, page 98*.
- Added BusProtocol & RequestLatency to the default Bus Parameter view. See *Parameter Views, page 98*.
- Added WorkBufferSize to the default VPM Parameter view. See *Parameter Views, page 98*.

Enabled Sorting in Versions Dialog Box

The list of tools and models (displayed in the Versions dialog box) can be sorted by Path or by Version number. See *Displaying CoMET and Tools / Models Versions, page 41*.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

New in CoMET 5.12

VaST VPM Transformer Toolset

Users of CoMET 5.12 have the ability to transform a VPM using the VaST VPM Transformer toolset.

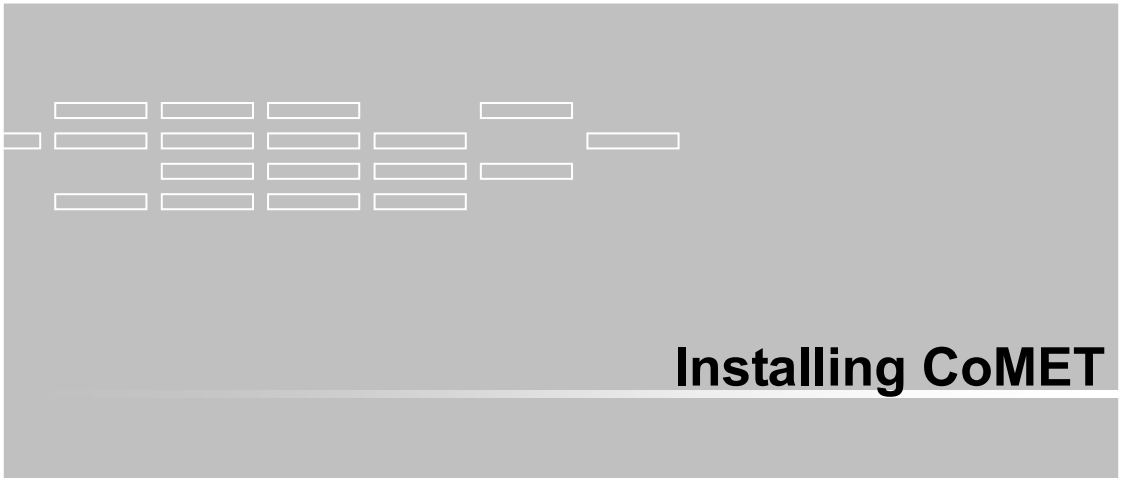
The VaST VPM Transformer is a toolset and methodology which allows users to modify the instruction set of a VaST Seed VPM. The VaST VPM Transformer Toolset needs to be purchased and installed before a VPM Transformer project can be started.

New in CoMET 5.11

Visual C++ Version 8 Supported

CoMET 5.11 supports Visual C++ Version 8. See *Tool Locations Configuration Settings, page 31*.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.



Installing CoMET

This chapter describes how to install CoMET on the Windows XP® and Windows 2000 operating systems.

CoMET Components

A CoMET release contains VaST Systems Technology's CoMET System Engineering Toolset including:

- The CoMET System Engineering Environment (CoMET), the graphical interface that provides a user-friendly and efficient design and analysis environment
- VaST's Architectural Modeling Programming Interface functions (AMPIs) – Virtual System Modeling function libraries
- VaST VPMs – one or more virtual processor models
- Module Library – a set of complete hardware and software example projects that you can use as references or as templates for your own projects

Installing CoMET on Windows XP or Windows 2000

The minimum system requirements are:

- Processor: IBM compatible PC with a Pentium or better processor
- Operating System: Windows XP or Windows 2000
- RAM: Minimum 64 Mbytes of DRAM
- Hard Disk: 800 Mbytes (with all supporting GCC cross-compilers)
- Video: A S-VGA video graphics adaptor
- A pointing device, e.g. mouse

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

NOTE: It is recommended that you reserve at least an additional 50 Mbytes of disk storage for swap space. You will require more swap area if you wish to simulate large designs.

To install CoMET on Windows XP / Windows2000:

- Insert the CoMET CD-ROM, or unzip the supplied zip file.
- If the installation program does not run automatically, start the installation program by selecting **setup.exe**, and follow the procedure displayed by the setup program. There is no need to reboot your computer after the setup procedure has completed.
- Copy the VaST license file (**license.dat**) from email or floppy disk (as provided) to the %VAST5_HOME%\license directory. (**%VAST5_HOME%** is the default installation directory where this release is installed.)
- Install the FLEXlm license manager software (See Globetrotter Software Inc.'s documentation for more information.).

NOTE: CoMET requires MsXml4 SP2. When CoMET launches it determines whether MsXml4 is installed. If not, CoMET launches the MsXml4 installer. You need to complete this installation before proceeding with CoMET.

NOTE: If you do not have the license files, or if licenses have expired, please contact VaST at: support@vastsystems.com.

Setting up Environment Variables

CoMET uses environment variables to define the location of CoMET components. When you install CoMET, these variables are set to default locations.

Usually, you will not have to change the settings of the environment variables from their default values, however, if you change the location (directories) of the CoMET files, you must update the environment variables so that CoMET can locate the associated files.

Environment Variables

CoMET uses the following mandatory environment variables:

- VAST5_HOME — The home directory of the CoMET System Engineering Environment (SEE) and associated system tools. CoMET automatically sets the local path to tools based on tool locations.
- LM_LICENSE_FILE — The location of the FLEXlm license manager license file.

NOTE: If you are using any third-party tools with CoMET (compilers, debuggers, etc), you may also require other environment variables, depending on the application. Refer to the documentation provided with the third-party tools for setup requirements.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Changing Environment Variables in Windows XP or Windows 2000

To change (or add) an environment variable in Windows XP or Windows 2000:

- From the **Taskbar**, click the **Start** button
- Select **Settings** / **Control Panel**
- Select the **System** icon
- In System Properties, select the **Advanced** tab
- Click the **Environment Variables** button
- Select the user environment variable to change
- Click the **Edit** button
- Type a new value for the variable (typically a path)
- Click **OK**.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Installing CoMET on a Network

Subject to licensing, CoMET may be installed in a single shared network location and used by multiple users. The procedure for installation is as follows:

Preparation

- Determine an appropriate shared directory in which CoMET can be installed. e.g `\\server1\shared`
- Determine an appropriate drive letter to which all user workstations can map the shared CoMET installation directory. e.g. `W:`
- Determine a home directory for CoMET under the shared directory, e.g. `vast5_home`

CoMET Installation

- Set up a workstation with the selected drive letter mapped to the selected shared folder. E.g, using the examples above, map `\\server1\shared to W:`
- From the workstation, install CoMET in the selected home directory on the mapped drive. Using the examples above, install CoMET in `W:\vast5_home`
- Run CoMET once to ensure automatic path updating takes place.

Copying a CoMET Installation

Alternatively you can copy a CoMET installation from another location. In this case, the procedure is:

- Copy the CoMET installation to the selected home directory on the shared drive.
- Set up a workstation with the selected drive letter mapped to the selected shared folder. E.g, using the examples above, map `\\server1\shared to W:`
- Ensure that the VAST5_HOME environment variable is set to the CoMET installation directory via the mapped drive. Using the examples above, set VAST5_HOME to the value `W:\vast5_home.` See *Setting up Environment Variables* above.
- Run CoMET once to ensure automatic path updating takes place. This updates the DTD path in the files %vast5_home%/Config/Vast6_1.0.0.cfx and Vast7_1.0.0.cfx. The DTD paths must agree with the CoMET installation. For example, if you have mapped the shared folder to V: and installed in the subfolder vast5_home, the second line of each of the files should read as follows:

  ```
  <!DOCTYPE Vast SYSTEM "V:\vast5_home\Config\Vast_v1.1.0.dtd">
  ```

  You may edit the DTD paths manually if you wish.

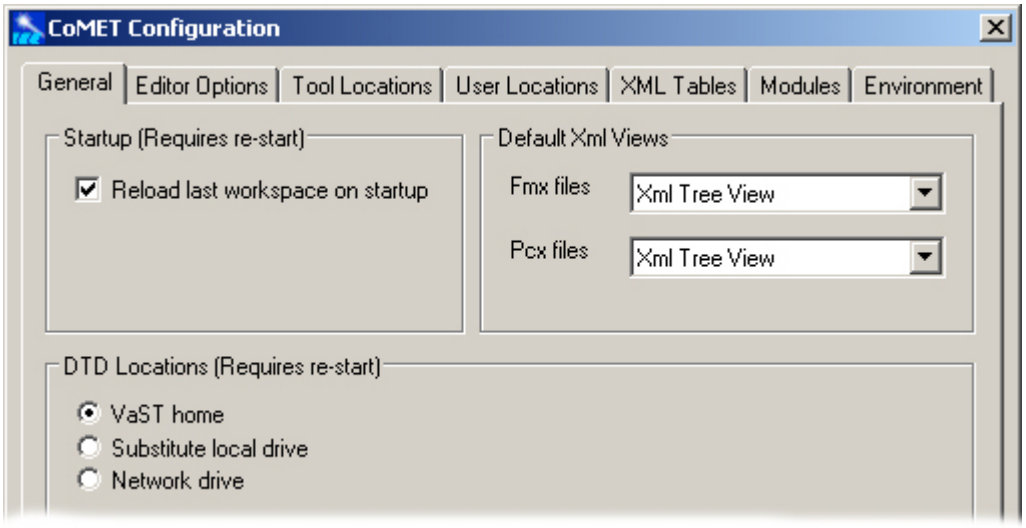
VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

On each Workstation

- Map the selected drive letter to the selected shared folder on workstations of all users. Using the examples above, map `\\server1\shared to W:`
- On each workstation, ensure that the VAST5_HOME environment variable is set to the CoMET installation directory via the mapped drive. Using the examples above, set VAST5_HOME to the value `W:\vast5_home`
  This can be set up in the COMET Configuration dialog (choose Tools/CoMET Configuration), Environment tab. Alternatively you can set it using standard Windows administration procedures. See *Setting up Environment Variables* above.

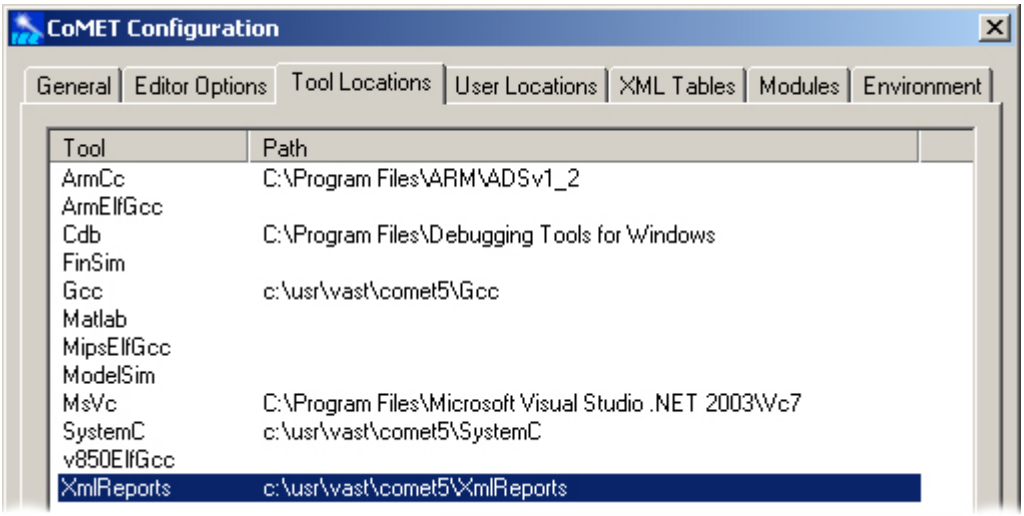
Set DTD Locations

- In the CoMET Configuration dialog (choose Tools/CoMET Configuration), General tab, set the DTD Locations to VaST Home. This results in a shared Config directory on the CoMET installation mapped drive. See *DTD Locations, page 27*.



Set XML Reports Locations

The XML Reports directory should be distinct for each user, to avoid overwriting other user reports. The XML Reports location specified in the CoMET configuration is set per workstation. Copy the %vast5_home%/XMLReports directory to a distinct location for each user. On the user workstation, specify the XMLReports location with the Tools/CoMET Configuration, Tool Locations tab.

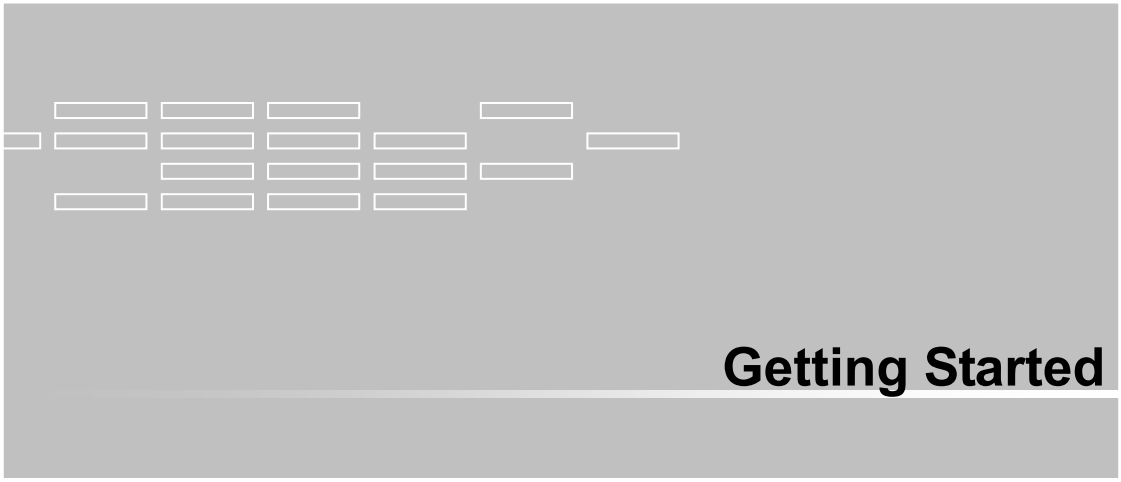


Uninstalling CoMET

To uninstall CoMET:

- From the **Taskbar**, click the **Start** button
- Select **Settings** / **Control Panel**
- Select the **Add/Remove Programs** icon
- Select **VaST Release xx-xx CoMET**
- Click the **Remove** button.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.



Getting Started

CoMET runs on the Windows XP® and Windows 2000 operating systems. Note that if CoMET has not been installed correctly, the necessary environment variables may not be set to the correct values, and you may experience difficulties running CoMET.

Starting CoMET in Windows XP/2000

To start CoMET in Windows XP/2000:

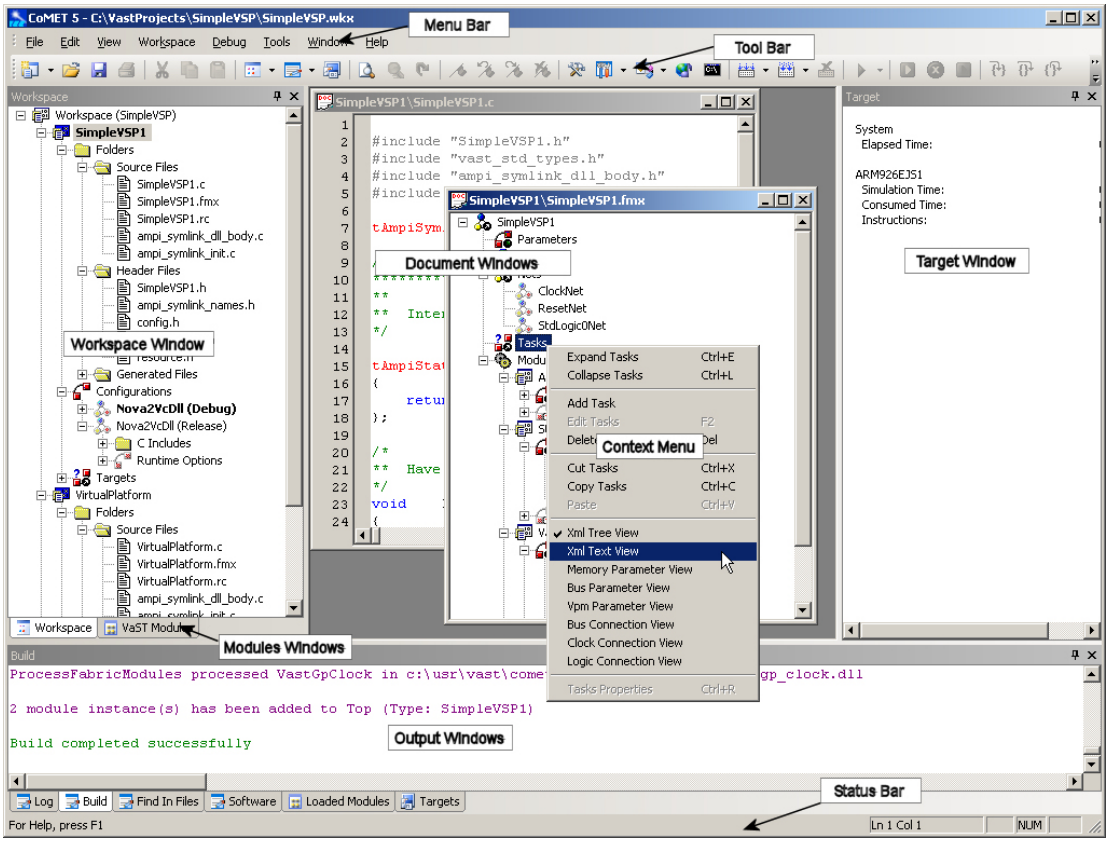
- From the **Taskbar**, click on the **Start** button
- Select **Programs / VaST / CoMET5 SEE**

or

- Double click on the desktop icon **CoMET5 SEE**

The CoMET System Engineering Environment

CoMET provides a graphical System Engineering Environment (SEE) in which you can build Virtual System Prototypes for analysis and testing.



CoMET SEE (System Engineering Environment)

The CoMET System Engineering Environment contains the following elements.

Menu Bar

The menu bar displays the commands you use to build, debug and simulate a Virtual System Prototype. Besides the standard **File**, **Edit**, **View**, **Window** and **Help** menus, additional menus such as **Workspace** and **Debug** provide access to utilities specific to CoMET.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Tool Bar

The tool bar provides quick access to commonly used commands. You click on a button in the tool bar to carry out the action represented by that button. Some tool bar buttons have a drop-down arrow to the right of the actual button. Click on the drop-arrow to display options associated with the button.

If you move your mouse pointer over a button, a *tool tip* (a small text window next to the mouse pointer) displays the purpose of that button. A short description also displays in the status bar at the bottom of the window.

See *Toolbar Buttons, page 17*, for additional details on some of the buttons in the Toolbar.

Status Bar

The status bar displays messages and information while you are building, debugging and simulating a Virtual System Prototype. The status bar is located at the bottom of the window.

Workspace Window

The workspace window displays the project files, header files, and other files associated with the workspace. You can use this pane to add files, and to select files for editing.

Target Window

Displays output information from the target application software when the Virtual System Prototype is executing. The target window also displays timing information about the simulation when the Virtual System Prototype is executing.

Output Windows

The output windows display messages during CoMET operation. These are color coded:

- **Gray**: CoMET information
- **Green**: CoMET success message
- **Dark Yellow**: Warning message from external tool
- **Red**: Error message from external tool
- **Black**: Output from external tool.

Document Windows

The Document windows display a variety of files in a variety of views. For example, an XML document such as a fabric module definition file can be displayed in XML Text view, XML Tree view and various Table views. You can choose the view from the Context menu. Open documents are listed in the Window menu. Document windows can edit source files, fabric module definition (.fmx) files, and display dll files and other project files.

Context Menu

Right click to display a context menu with options appropriate to the context in which you clicked. For example, right click on a document window to view document properties and actions, or choose view format from the document context menu.

Modules Window

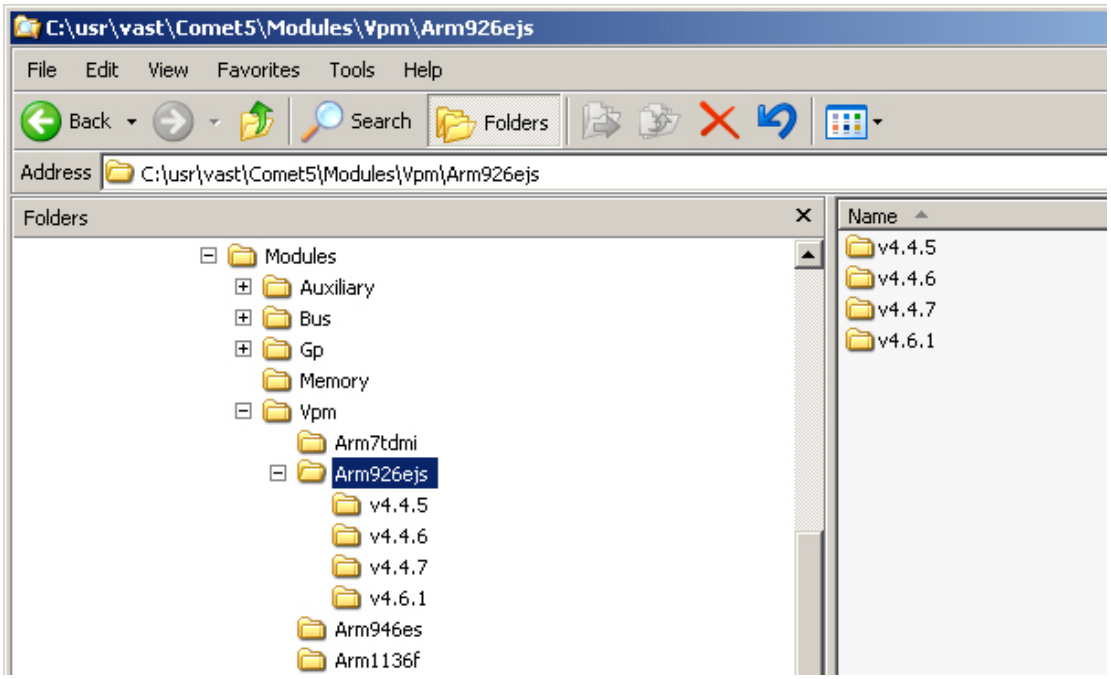
The Modules Windows lists available modules. You can add modules to projects by drag and drop or copy and paste from the Modules Window to a fabric module definition (.fmx) file in a Document Window. To refresh the contents of the module window, choose **Tools/Rescan Modules**.

Multiple Versions of Models within CoMET

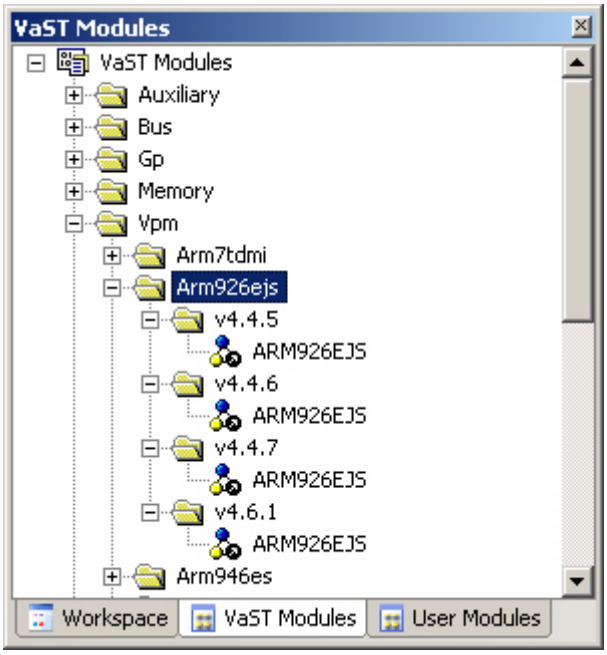
CoMET supports multiple versions of a model to be located in the modules (%VAST5_HOME%\Modules) directory. To enable multiple versions of a Model to co-exist in the modules directory, the CoMET directory structure includes a set of sub-directories under each Model type. The sub-directory under each Model type identifies the Model. The sub-directories under the Model type directory each contain a different version of the Model, with the directory name reflecting the version of the model.

The screen capture below shows how different versions of the ‘Arm926ejs’ VPM are organized in the ‘Modules\Vpm’ directory.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.



The screen capture below shows multiple versions of the 'Arm926ejs' VPM as it appears in the VaST Modules window in CoMET.



Maintaining Multiple Versions of a Module CoMET

VaST's 'Models and Platforms' releases install only the latest version of all Models distributed as part of VaST's library of Models (VPM, Standard Bus, GP Memory etc.) The Models are placed in a version sub-directory under the '%VAST5_HOME%\Modules*Module_Type**Module_Name**Version_Number*'.

When installing the latest Models and Platform release, if you wish to maintain earlier versions of your Models, follow the steps described below:

1. Backup the '%VAST5_HOME%\Modules' directory.
2. Follow the installation instructions documented in the Models and Platforms release notes for installing the release.
3. Within your Backup location, for each Model version you wish to retain, create a sub-directory which includes the Model name and Model version number. For example, for v4.5.7 of an ARM926E VPM the path of the newly created sub-directory would be as follows:
   '%VAST5_HOME%\Modules*VPM\ARM926E\v4.5.7*'
4. Copy the model sub-directory (as detailed in step 3 above) from the backup location to the '%VAST5_HOME%\Modules*Module_Type**Module_Name*\' directory.

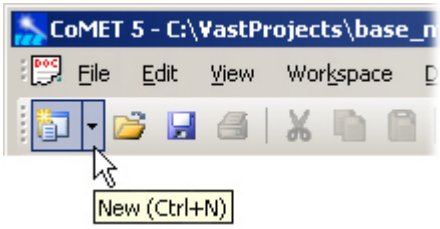
For further information on multiple versions of models with in CoMET see *Maintaining Multiple Versions of same Model in CoMET*.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Toolbar Buttons

The tool bar provides quick access to commonly used commands. You click on a button in the tool bar to carry out the action represented by that button. The following sections describe some of the special buttons on the Toolbar.

"New File" Button

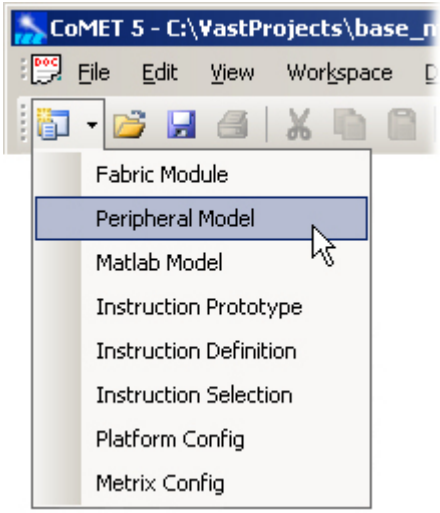


You can use the "New File" button to create a new:

- Fabric Module
- Peripheral Model
- Matlab Model
- Instruction Prototype (requires the VaST VPM Transformer toolset)
- Instruction Definition (requires the VaST VPM Transformer toolset)
- Instruction Selection (requires the VaST VPM Transformer toolset)
- Platform Configuration
- Metrix Configuration

To create a new file:

- From the **Tool Bar**, click the down arrow next to **New File** button

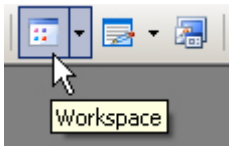


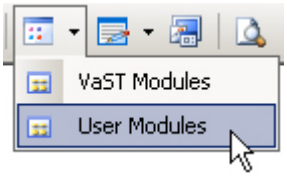
- Select the type of file from the drop down menu.

NOTE: Clicking the New File button will automatically create a new text file.

Workspace Button

If the Workspace is not visible on the CoMET SEE click the Workspace button on the Tool Bar to display the Workspace.

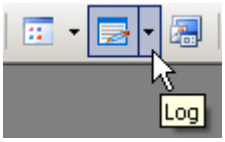




You can use the drop-down arrow on the right of the Workspace button to display the VaST Modules and User Modules windows.

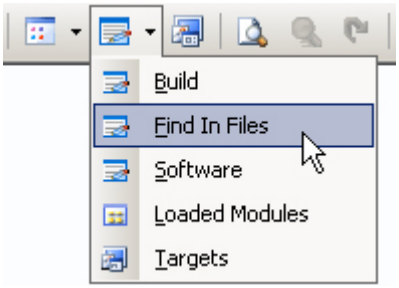
Log Button

The **Log** button allows you to view, open and switch between the output windows.



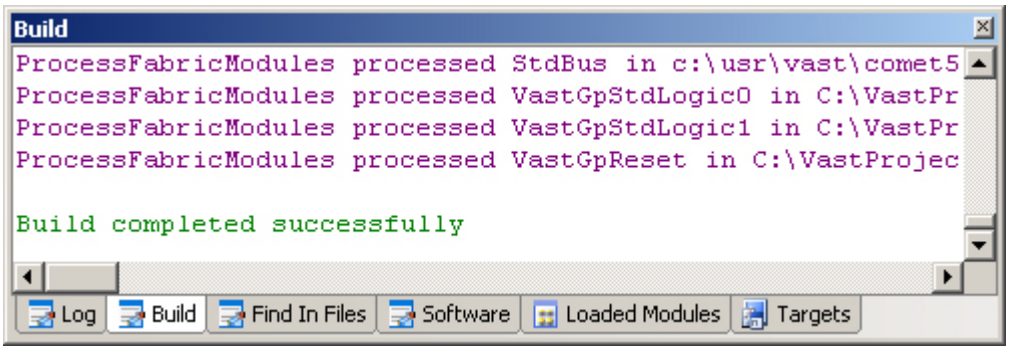
To open or switch between the output windows:

- Click on the **Log** button to open the Log output window.
- To open the other output windows click the down arrow next to **Log** button



- Select the output window you wish to open or switch to from the drop down menu.

The following screen shot shows all the output windows arranged as separate tabs.

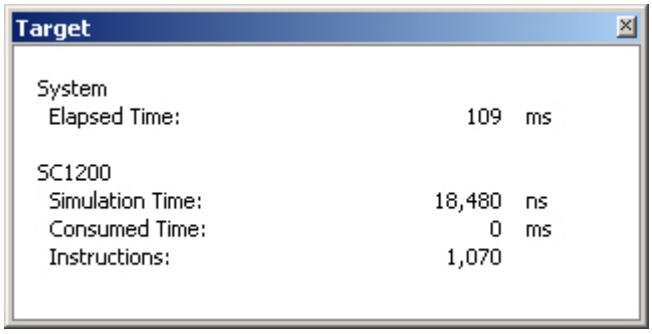


VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Target Button

If the Target window is not visible on the CoMET SEE click the **Target** button on the Tool Bar to display the Target window.

The following screen shot shows the Target window.



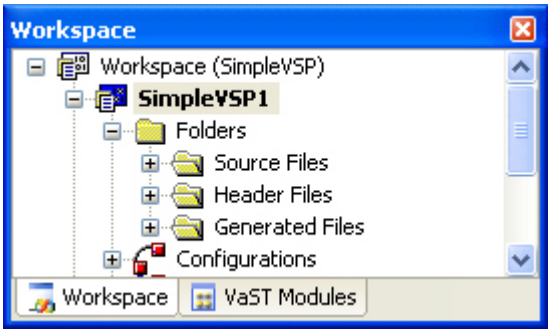
Doing the Tutorials

The tutorials provide a step-by-step introduction to developing systems in CoMET. See the *CoMET Tutorial*. This is a good point to work through the tutorials. The remainder of this guide is a reference document. You can read the sections in any order.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

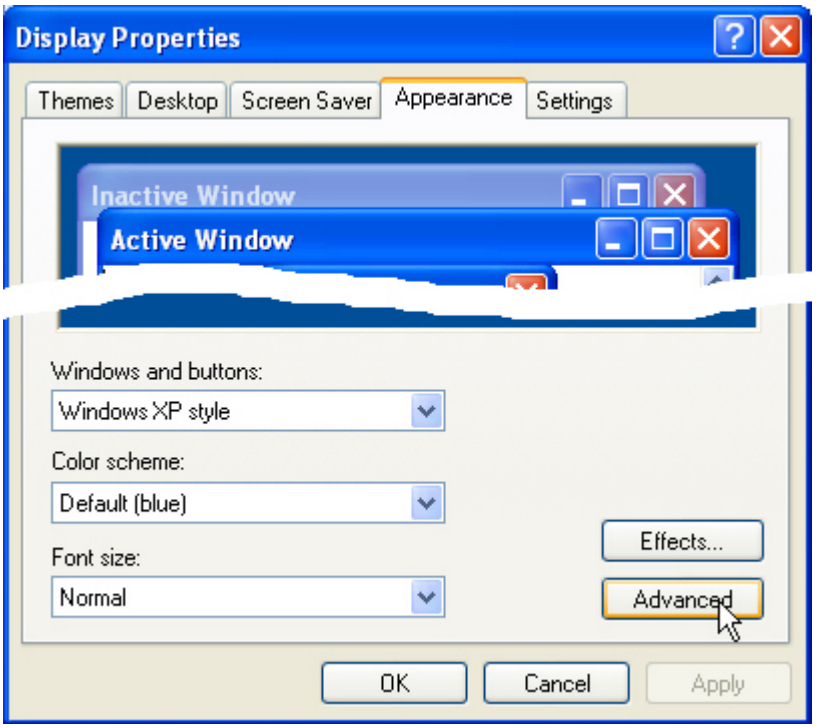
Windows XP Theme Display Issue

When using Windows XP Theme display options, selected tabs may be indistinguishable from unselected tabs. For example, in the diagram below the Workspace tab is very similar to the VaST Modules tab. This is due to Windows XP color scheme where the two color options are very similar.



To address this issue you can alter the shade of the unselected tab. The following procedure outlines how you can change the shade of unselected tabs:

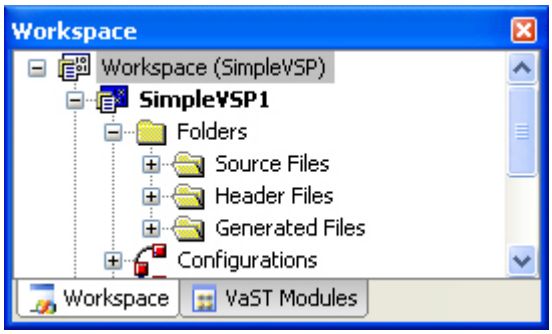
- From the **Taskbar**, click on the **Start** button, and select **Control Panel**
- On the **Control Panel** window double-click **Display**
- On the **Display Properties** dialog box select the **Appearance** tab
- Click the **Advanced** button to select advanced options



VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

- On the **Advanced Appearance** dialog box select **3D Objects** from the **Item** drop-down list
- From the **Color 1** drop down list select a darker color than one that is already selected.

- The following diagram shows a selected tab (Workspace tab) and an unselected tab (VaST Modules), with a darker color selected for the unselected tab.



Arranging Windows

The CoMET System Engineering Environment (SEE) contains two basic window types: tool windows and document windows. These two window types behave in slightly different ways. You can increase the viewing and editing space for code, depending on how you arrange the windows in the SEE.

Document windows

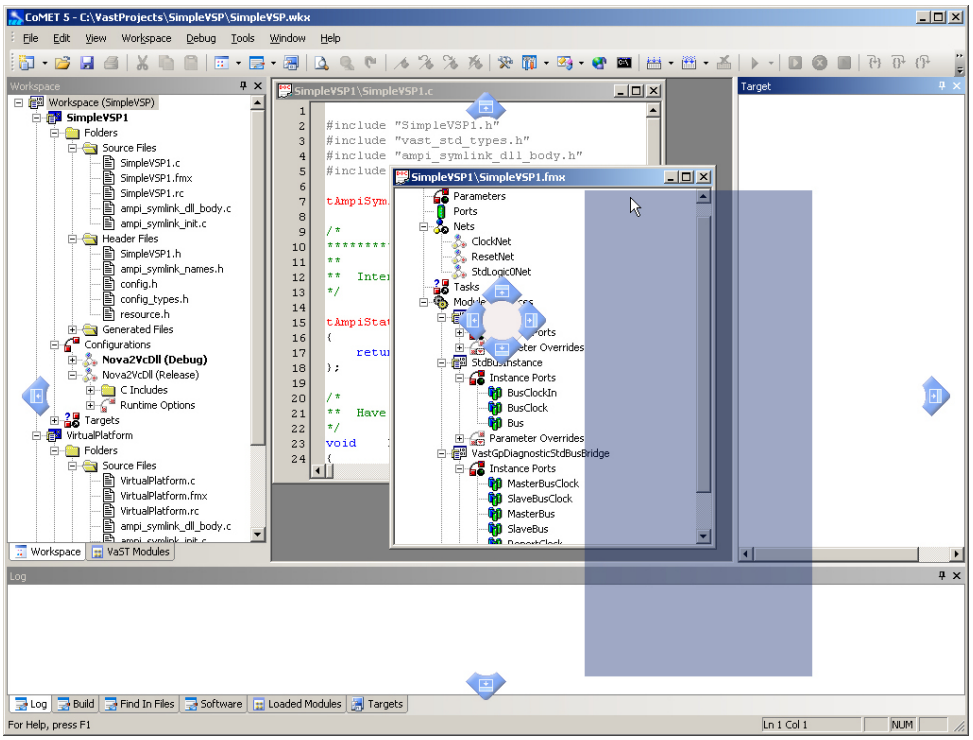
Document windows are dynamically created when you open or create files or other items. The list of open document windows appears in the Window menu in the current z-order, with the top-most window listed first. You can arrange windows into cascading or tiled stacks.

Tool windows

Tool windows, which include the Workspace, Output and Modules windows, are listed on the View menu.

Docking Tool Windows

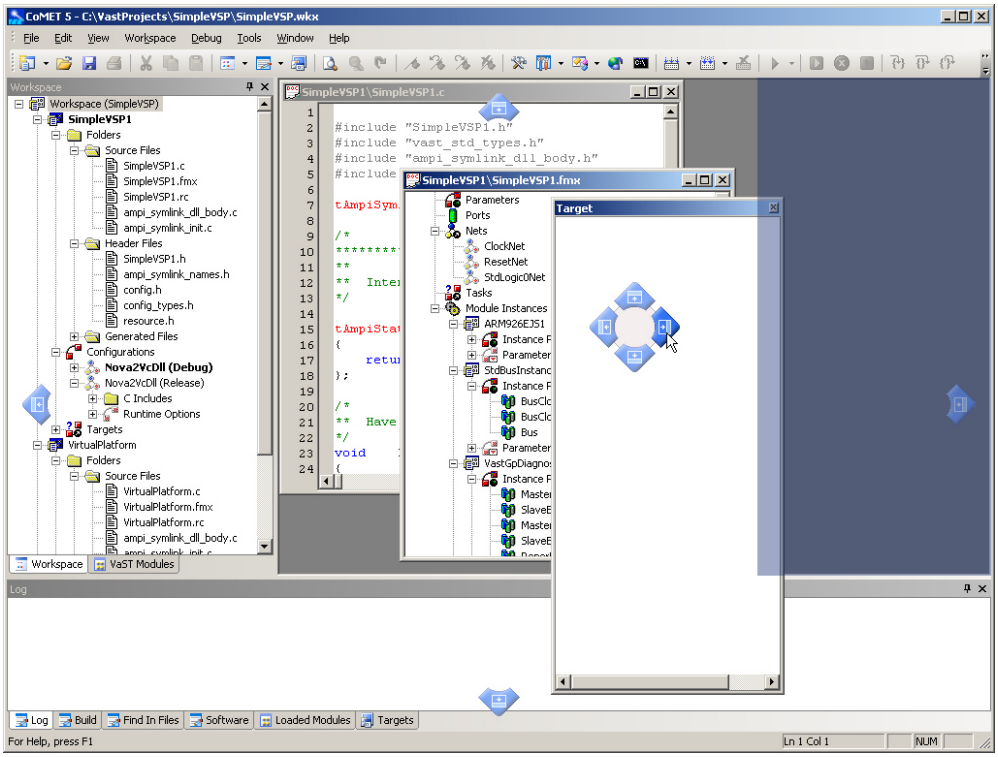
You can arrange the Workspace, Output and Modules windows to dock against the edges of the SEE or float over the SEE.
To un-dock a tool window, click in the title bar of the window and drag it to a new location within the SEE.



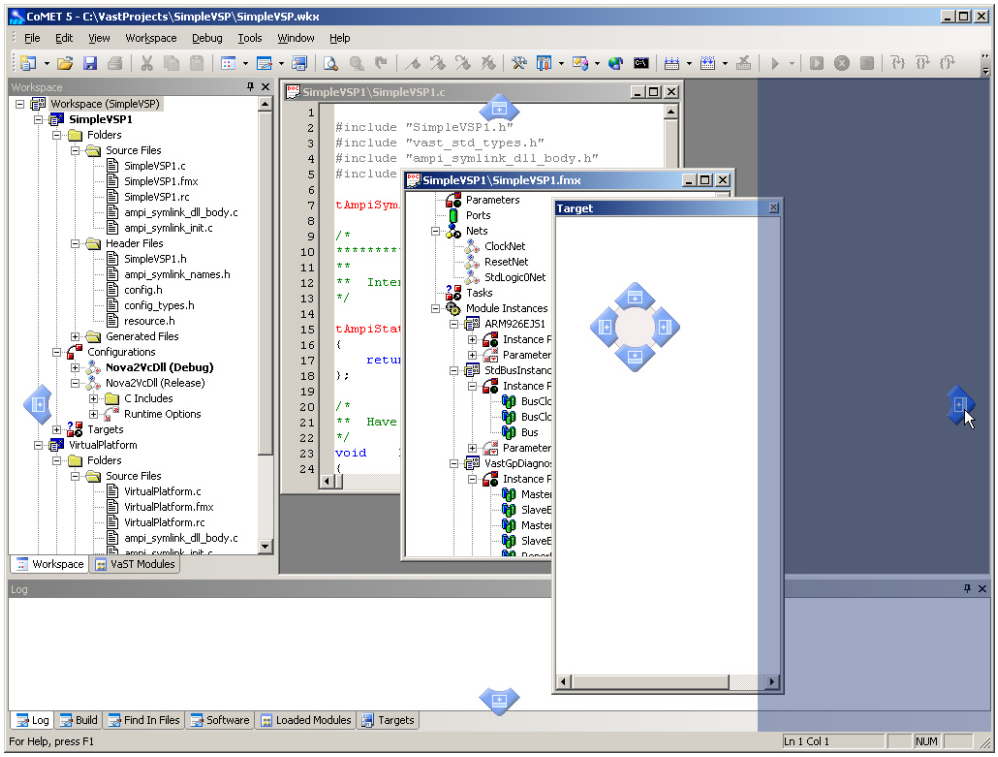
To re-dock the tool window, click in the title bar and drag the cursor to any of the available docking positions displayed. When the cursor is placed over a docking position, the resulting location of the window, if docked at that docking position, is shown by a shaded area.

The following diagrams show the same tool window (the Target window) being docked at different positions within the SEE.

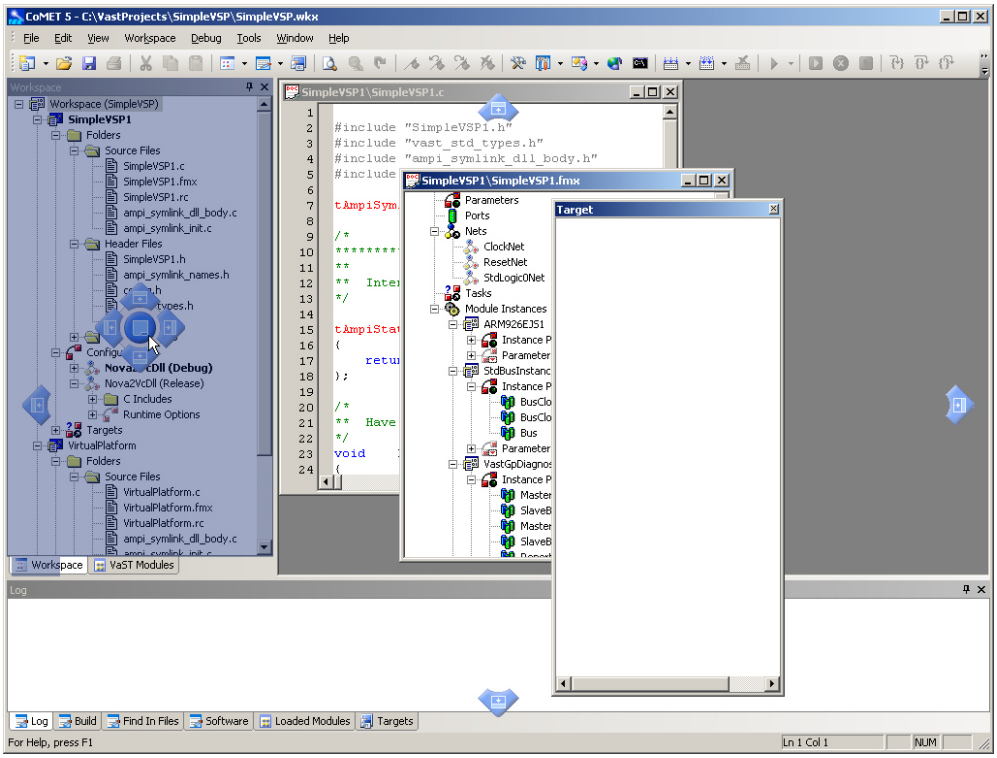
VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.



Docking on right hand side of the center pane



Docking on the right hand side of the SEE



Docking as a tab window into exiting docked windows on left hand side of SEE

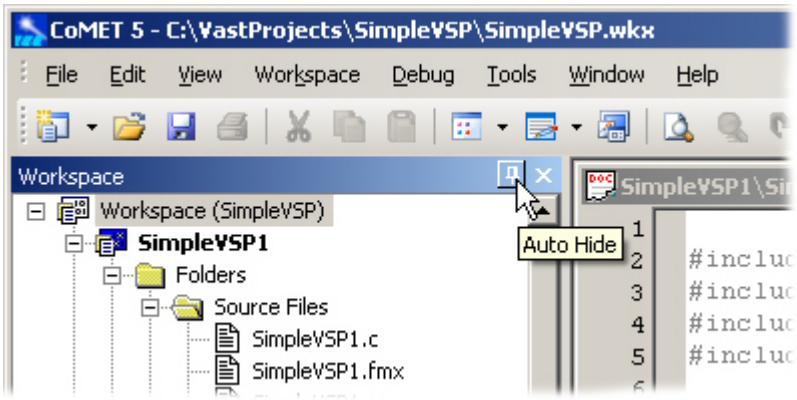
You can also double-click in the window's title bar to toggle between the docked and undocked position.

To undock a tools window from multiple windows docked together in tabs, double-click the tab of the window you wish to undock. Double-clicking the window's title bar of a multiple tabbed docked window will result in all windows being undocked.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

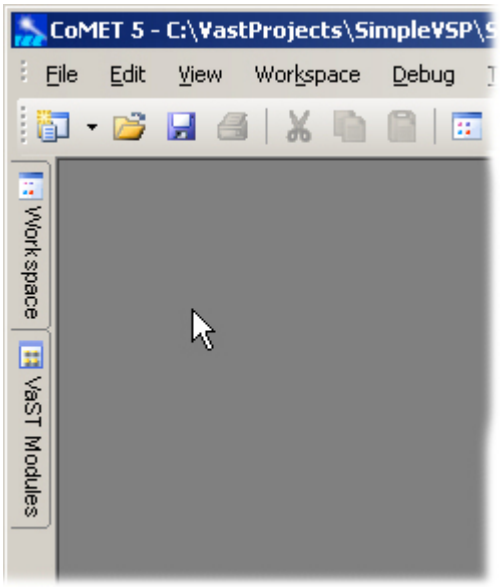
Auto Hide Button

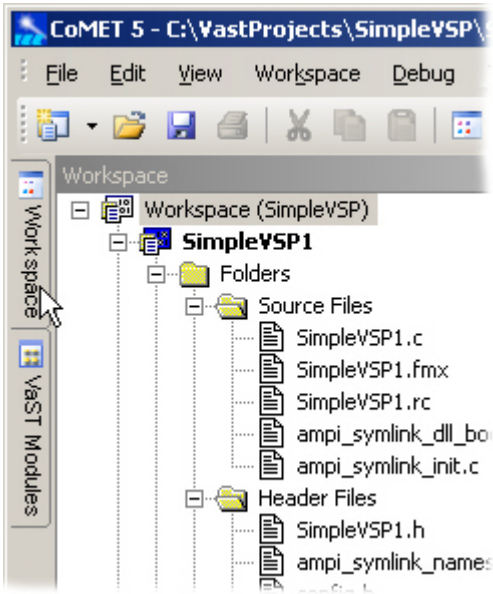
When a tool window is docked you have the option of selecting to automatically hide the window. To do this click the Auto Hide button on the right corner of the tool window.



Auto Hide button on a Tool window

To display an auto hidden tool window simply scroll the pointer over the hidden tool window tab.





To turn off the Auto Hide option on an auto hidden window click the Auto Hide button again.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Configuring the CoMET System Engineering Environment

You can modify the configuration of the CoMET System Engineering Environment to suit your preferences with the CoMET configuration settings.

To display the CoMET Configuration dialog:

- From the Tools menu, select CoMET Configuration

or

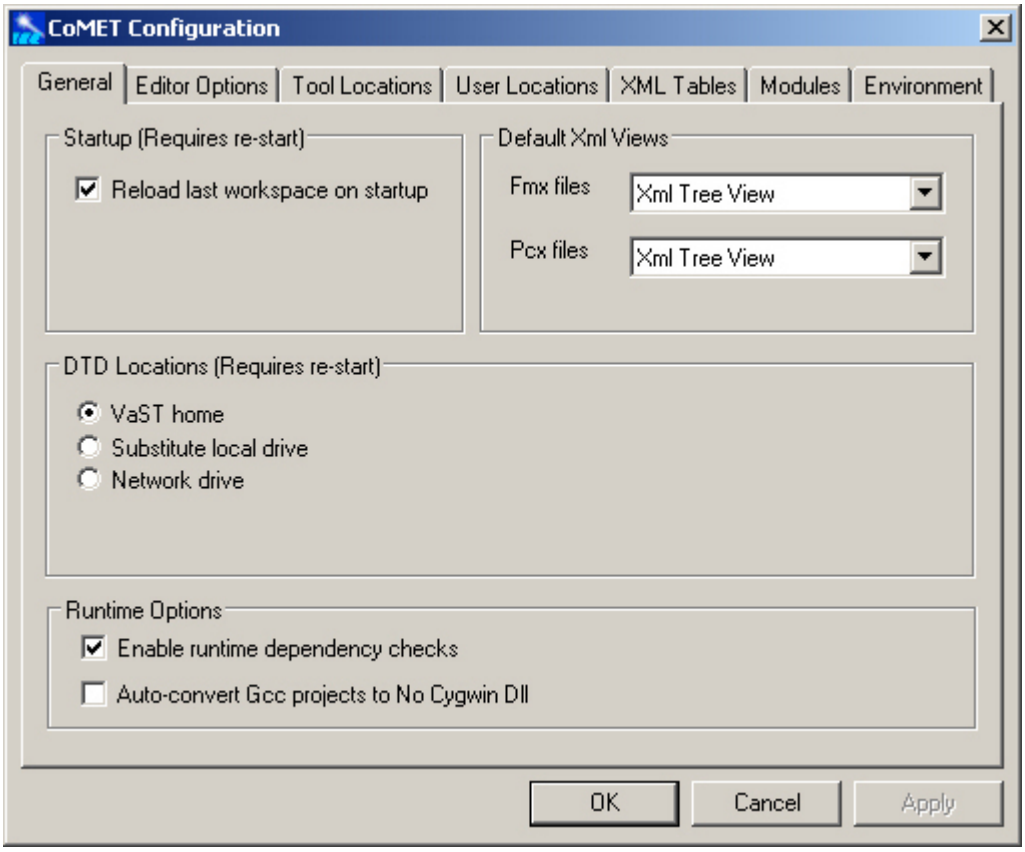
- Click the **CoMET Configuration** tool bar button

The Configuration Settings are defined on several tabs, described in the following sections. When you have made all your selections,

- Click **OK**

General Configuration Settings

These options allow you to customize the general behavior of the CoMET System Engineering Environment. Check the option to enable the feature. The following list explains the behavior of each option when selected.



General Configuration Tab

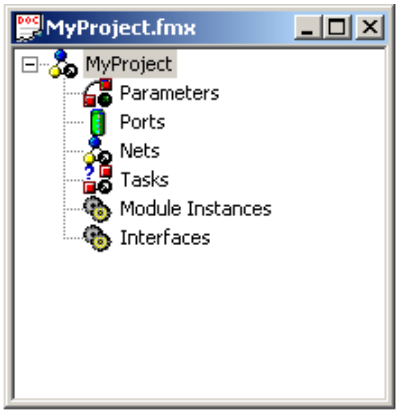
VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Reload last workspace at startup

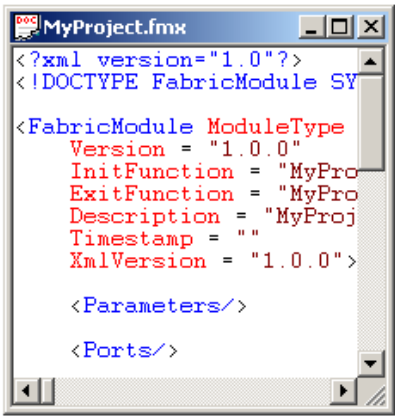
By default, the CoMET System Engineering Environment starts without a workspace. If you select this option, your last workspace is automatically loaded when the CoMET System Engineering Environment starts.

XML documents open in tree view by default

Determines whether XML configuration files are displayed as a tree view or in XML source.



XML in Tree View



XML in Source View

DTD Locations

These options are required for users collaborating on development using a source file repository. The Document Type Definitions that appear in the various XML files (e.g. .fmx, .pcx) must refer to an absolute file location. If these file locations differ from user to user there will be an XML source file conflict in the repository. These options provide a mechanism by which all users can refer to the same absolute file location, avoiding conflict.

Vast Home

If all users agree to install CoMET in the same local directory then the file locations agree and the default **Vast Home** setting choice is appropriate.

Substitute Local Drive

Otherwise, users can agree to map the DTD location to the same virtual drive by choosing the **Substitute Local Drive** option and selecting the agreed drive letter.

Network Drive

Alternatively, the DTDs may be placed on an agreed network drive, and all users can refer to it by the same UNC. Choose **Network Drive,** browse and select the agreed UNC.

Default XML Views

Choose the view you prefer as a default on first opening a .pcx or an .fmx file. For more on table views, see *XML Tables Configuration Settings, page 33* and *Using Table Views, page 97*.

Runtime Options: Enable runtime dependency checks

Check this box to enable runtime dependency checks. This is the default behavior. At the beginning of simulation, CoMET checks dependencies in all projects in the workspaces. CoMETs prompts to rebuild projects with out-of-date builds.

Uncheck this box to disable runtime dependency checks. CoMET performs no dependency checking. A warning appears at the start of simulation:

Warning: Runtime dependency checking will not be made

Runtime Options: Auto-convert Gcc projects to No Cygwin Dll

Check this box to automatically convert any Gcc project to use the no cygwin Dll option. With this option enabled:

- Loading an existing Gcc project into CoMET will automatically use the no cygwin option (`-mno-cygwin`).
- Starting a new Gcc project in CoMET will automatically use the no cygwin option (`-mno-cygwin`).

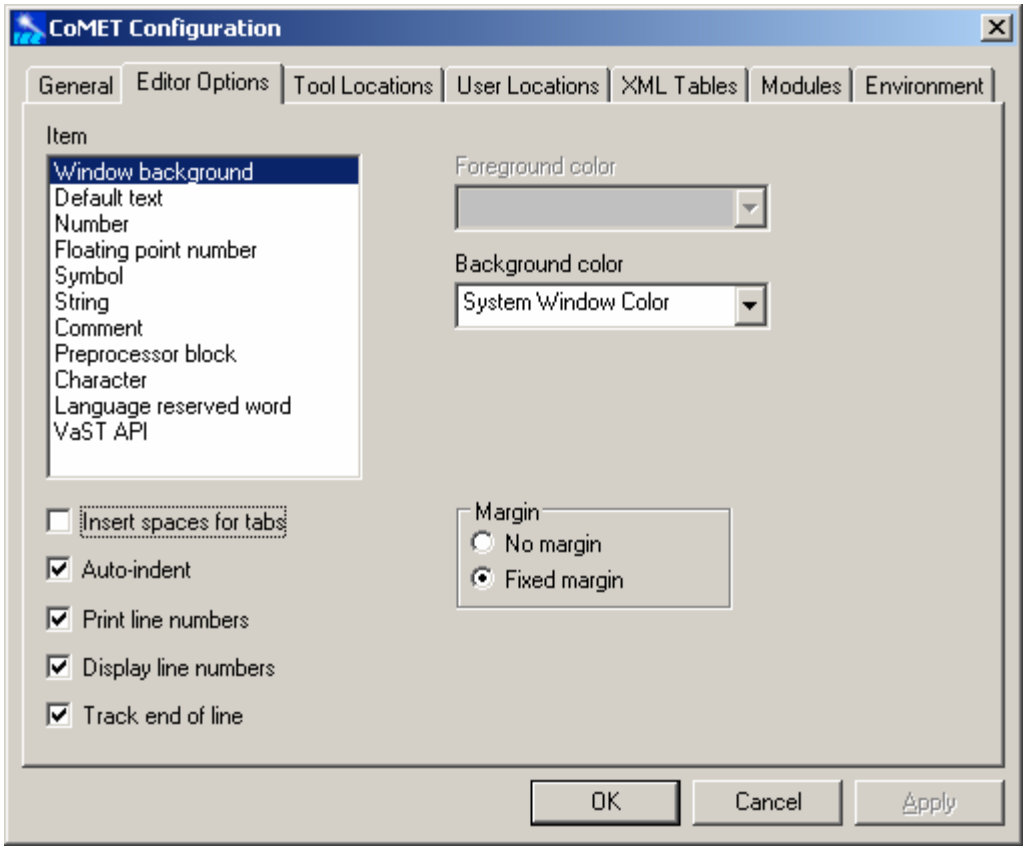
Fabric, VDag, Metrix and general Gcc project configurations are supported by this feature.

NOTE: It is also possible to enable/disable the No Cygwin Dll option when creating a new Cygwin Gcc project. Enabling the No Cygwin Dll option in the CoMET Configuration dialog box will override the No Cygwin Dll option in the New Project wizard. See *Adding a New Project, page 45*.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Editor Options Configuration Settings

These options allow you to customize the appearance and behavior of the text editor. Check the option to enable the feature. The following list explains the behavior of each option when selected.



Editor Options Configuration Tab

Item

You can set the colors used by the text editor for various fields, such as comments and numbers. Click on the item and select the color(s) from the Foreground/Background color drop lists.

Insert spaces for tabs

When you press the <tab> key, four <space> characters will be inserted, not a <tab> character.

Auto-indent

When you press the <enter> key, the indent of the new line will match the indent of the current line.

Print line numbers

Line numbers will be printed when you print a file.

Display line numbers

Edit windows will display line numbers on the left of the frame.

Track end of line

Prevents the cursor from continuing past the end of a line. For example, when you move a cursor that is at the end of a long line to the following (shorter) line (by using the down-arrow button), the cursor will go the end of the line instead of staying in the same column.

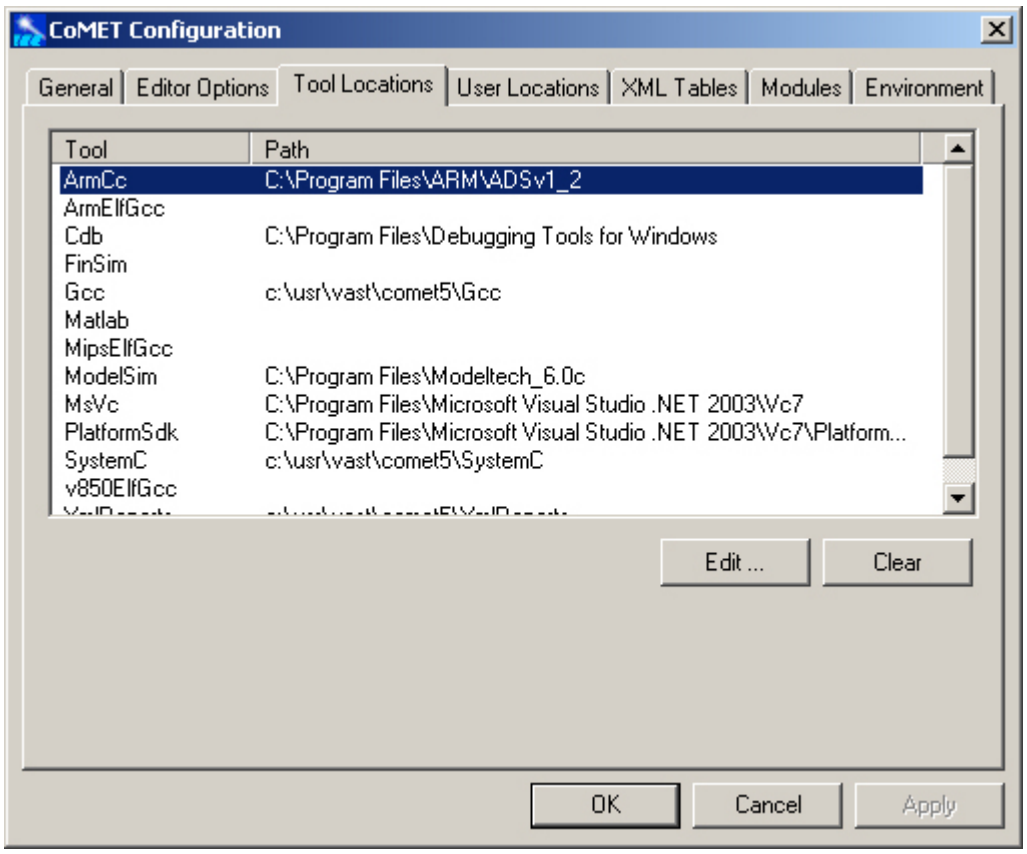
Margin

Selecting **Fixed margin** displays a gray margin on the left-hand edge of code windows. The margin displays symbols indicating bookmarks and breakpoints. If you select **No margin**, then highlighting of the line of code indicates bookmarks and breakpoints.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly
authorized in a written agreement with VaST.

Tool Locations Configuration Settings

You can specify the location of the ancillary tools that you use in conjunction with CoMET for building, debugging, and executing the Virtual System Prototype.



Tool Locations Configuration Tab

MsVc (Microsoft Visual C++): On CoMET installation, CoMET detects the most recent version of MS Visual C++. If version 8 (.net) is installed this is automatically detected and becomes the default value for MsVc. If you install CoMET over an earlier CoMET installation, it takes the existing setting for the MsVc parameter. You can change the version manually by altering the value of this parameter. CoMET supports MS Visual C++ versions 6, 7, 8 and 8-Express.

Gcc: Choose a version of the Gcc compiler appropriate to the tools and features you are using, such as VBus project support and FinSim support. CoMET supports Gcc 3.2 (set Gcc Location to (%vast5_home%/gcc) and Gcc 2.91.57 (Set Gcc location to $vast5_home/gcc_egcs_2.91.57). See *CoMET 4 and SystemC Compatibility, page 159*.

NOTE: If you have Cygwin installed on the same computer as CoMET, you must ensure that you use the version of Cygwin supplied with CoMET. Later versions of the cygwin1.dll do not work correctly with the version supplied with CoMET.

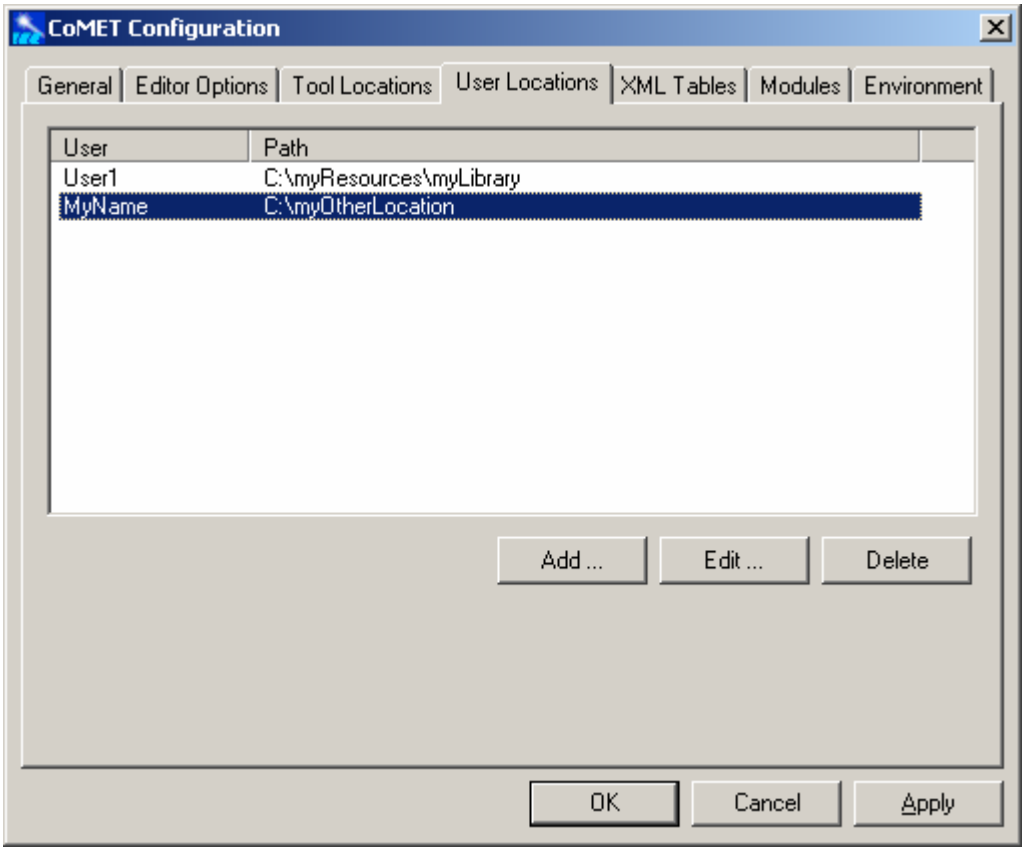
VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

User Locations Configuration Settings

The User Locations tab allows a user to configure variables to represent directory paths on the user's machine. CoMET substitutes the variable for the corresponding path on closing a project file or workspace file. On opening the project file or workspace file CoMET substitutes the path for the corresponding variable.

This is useful if you have libraries or other file resources outside the CoMET workspace, and you share the workspace between users who keep the file resources in a different directory configuration. If each user sets the user location variable to the correct directory path on that user's machine, then the workspace can be shared without the users requiring identical directory configurations.

It is not necessary to create User Locations for paths within the workspace. CoMET stores these as relative paths.



To create a new User Location, click the Add button.

To edit the User Location path, select the User Location entry and click the Edit button, or double click the User Location entry.

To alter the User Location name, select the User Location entry and press F2 or, with the User Location entry selected, click the name.

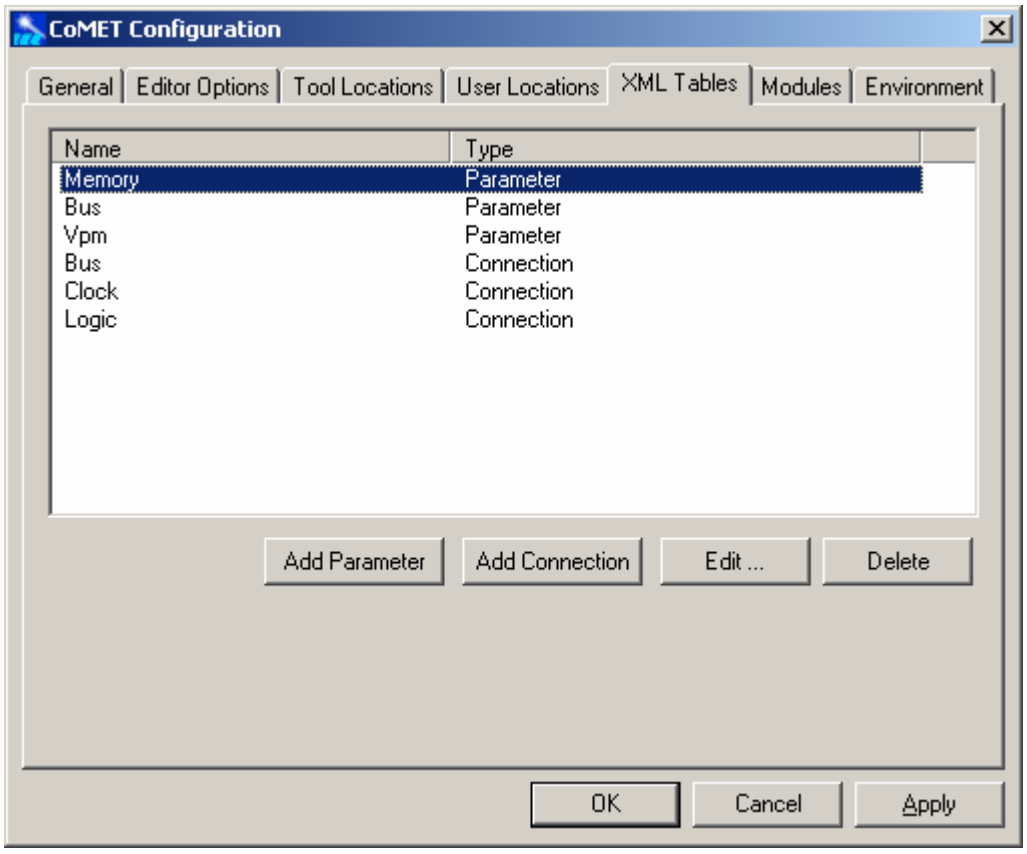
VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

XML Tables Configuration Settings

The XML Tables Configuration determines the layout of XML Table views within Document windows. These views are available from the right-click context menu for Fabric Module Definition files. They provide a tabular listing of selected elements within the module.

There are two types of table:

- Parameter - lists module instances with their module type and columns corresponding to specified parameter types.
- Connection – lists module instances with their module type and all connections of specified types

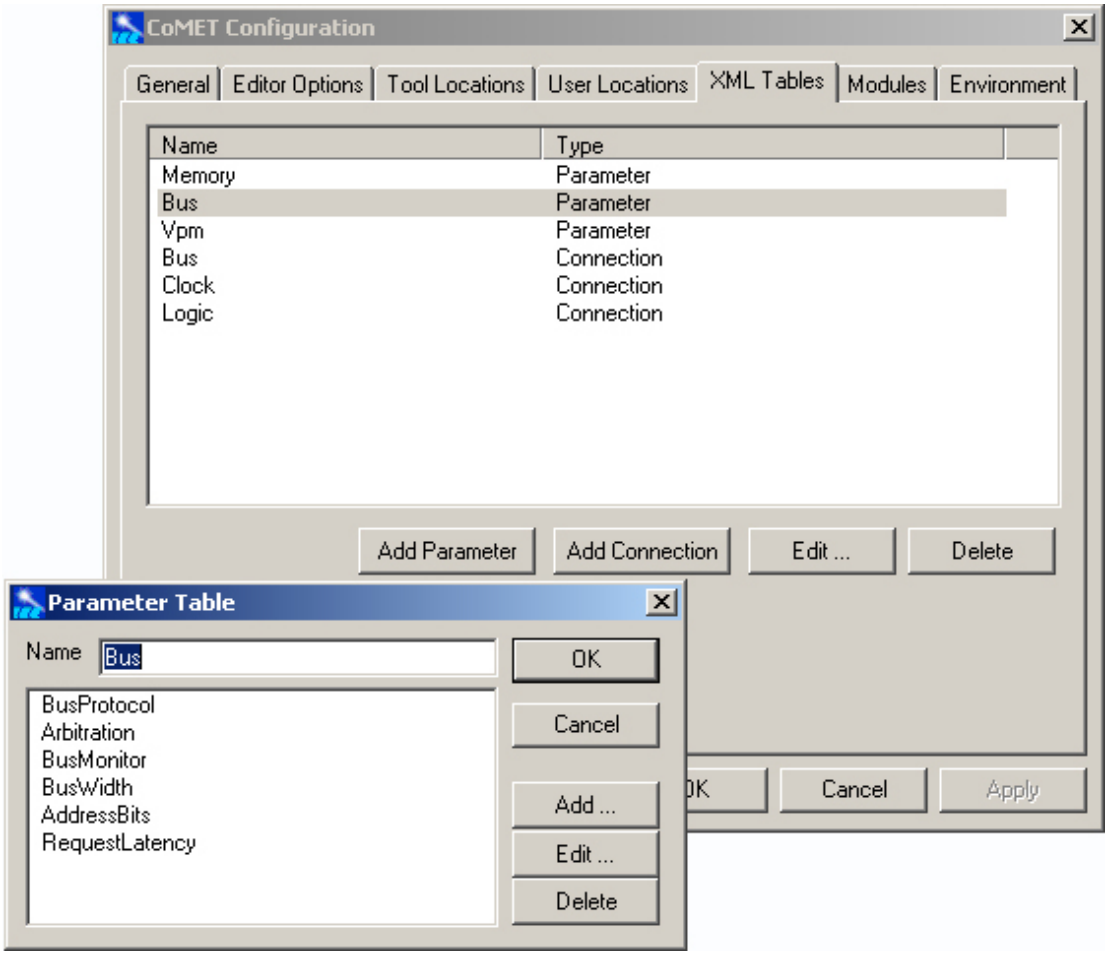


XML Tables Configuration Tab

Several parameter and connection views are provided with CoMET. You can edit the existing views or add new tables.

Parameter Table Editing

To create a new parameter table, click **Add Parameter** in the XML Tables Configuration tab. To edit an existing parameter table, select the table and click **Edit**, or double click the table.

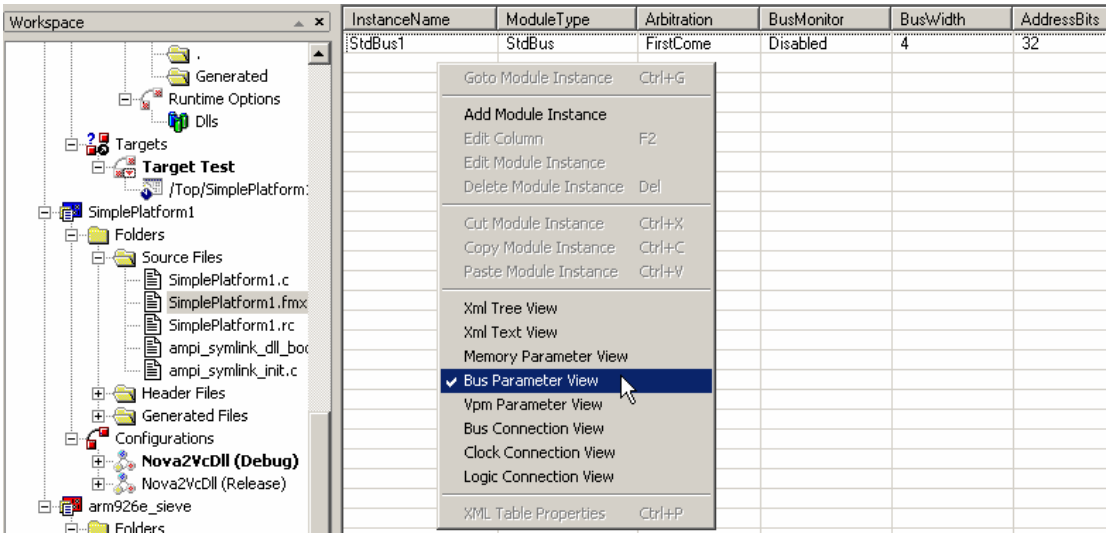


XML Tables Configuration Parameter Table Editor

In the Parameter Table dialogue, each of the listed parameter types corresponds to a column in the Parameter Table. Click **Add** to add a new parameter. Use the hierarchical parameter naming scheme to distinguish parameter types. See *Hierarchical Parameter Type Naming, page 76*.

Select an existing parameter and click **Edit** to change its name or Delete to remove it. Click **OK** to return to the XML Tables tab, and **OK** again to save changes. The corresponding view of the fabric module definition xml file is shown below.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

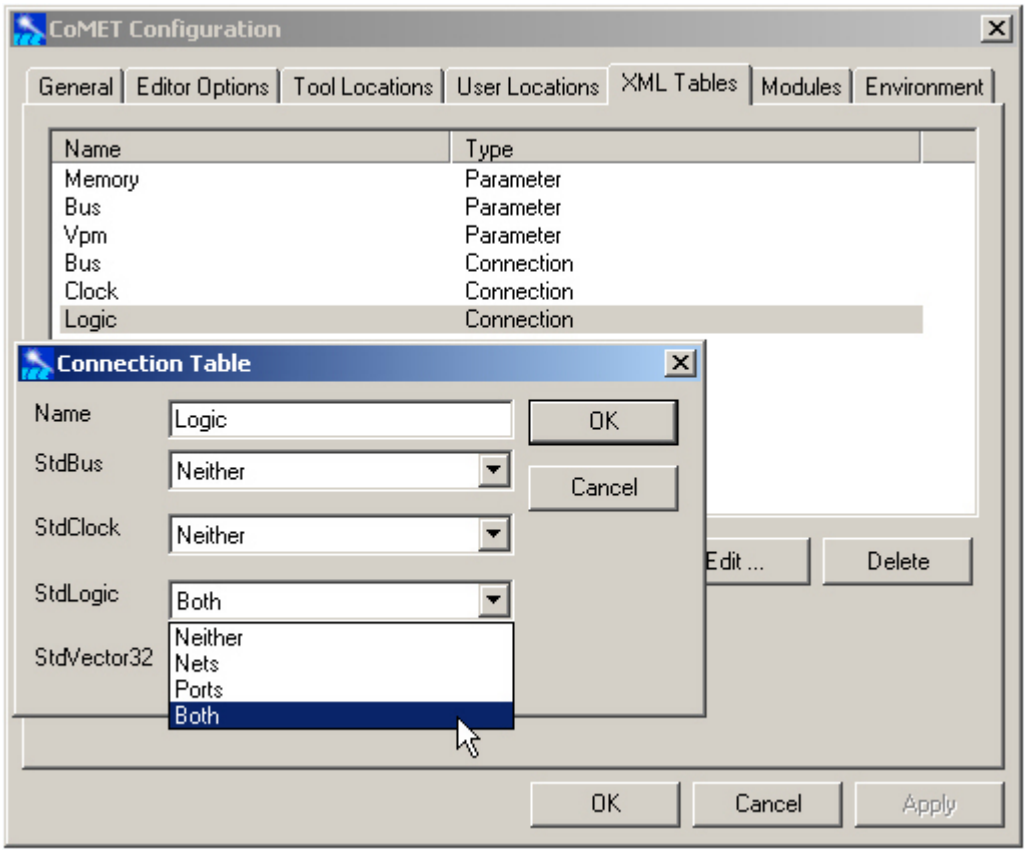


Corresponding Bus Parameter View

See also *Parameter Views, page 98*.

Connection Table editing

To create a new connection table, click **Add Connection** in the XML Tables Configuration tab. To edit an existing connection table, select the table and click **Edit**, or double click the table.

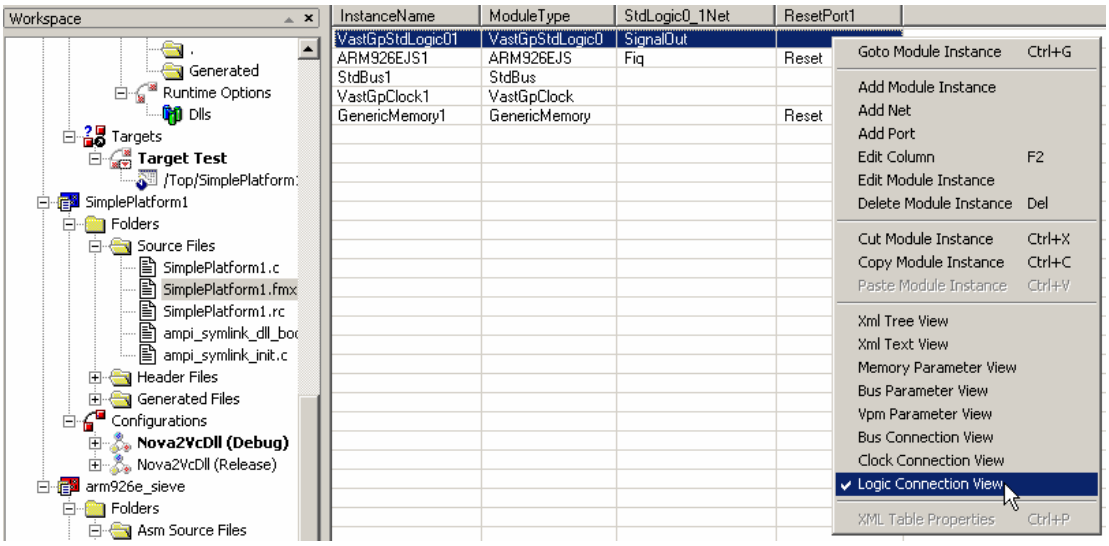


XML Tables Configuration Connection Table Editor

The Connection Table dialog allows you to specify the types of connection to be listed in the table. Available types are shown. For example, you can choose to display neither Nets nor Ports of type StdBus, but both Nets and Ports of type StdLogic.

The view shows rows with module instance name, module instance type, and additional columns each of which corresponds to a connection of one of the specified types. The corresponding view of the fabric module definition xml file is shown below.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

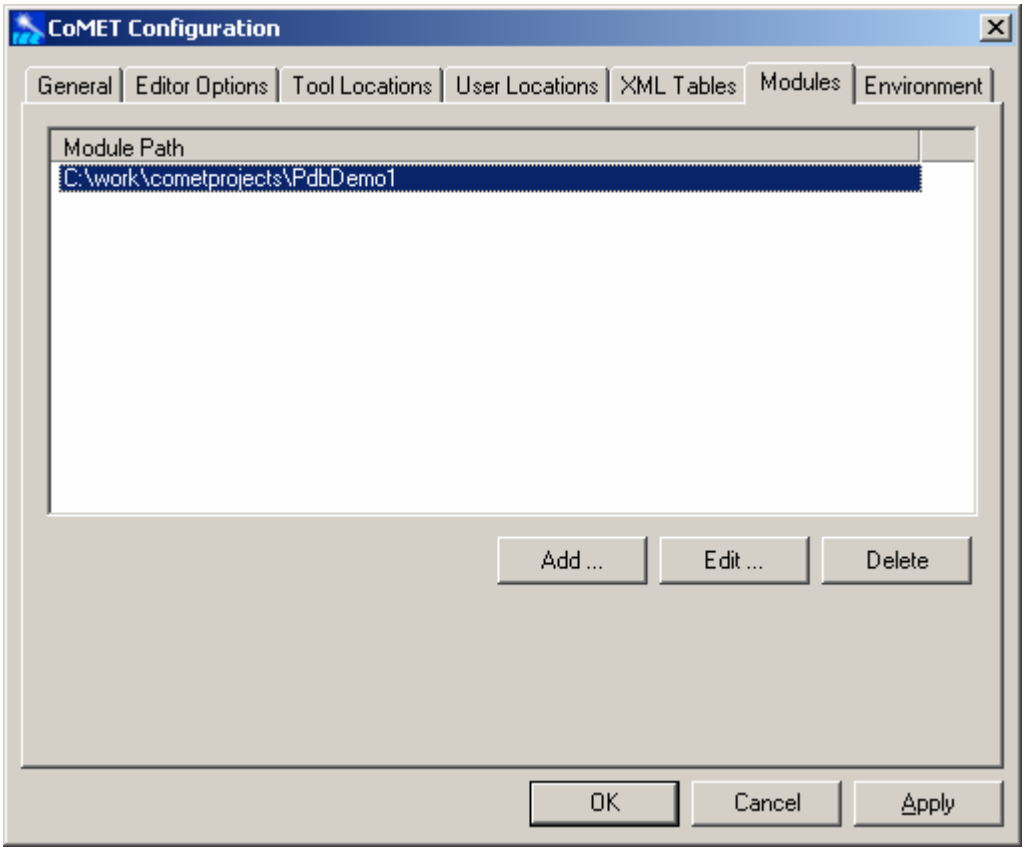


Corresponding Logic Connection View

See also *Connection Views, page 99*.

Modules Configuration Settings

CoMET comes with a preconfigured module library, arranged as a tree structure. You can view this library in the VaST Modules tab of the Modules window. You can add your own module libraries using the CoMET Configuration Modules Tab.



To add a library:

- Click **Add**
- Browse to the top level directory containing your library of module .dll files.
- Click **Apply** or **OK**

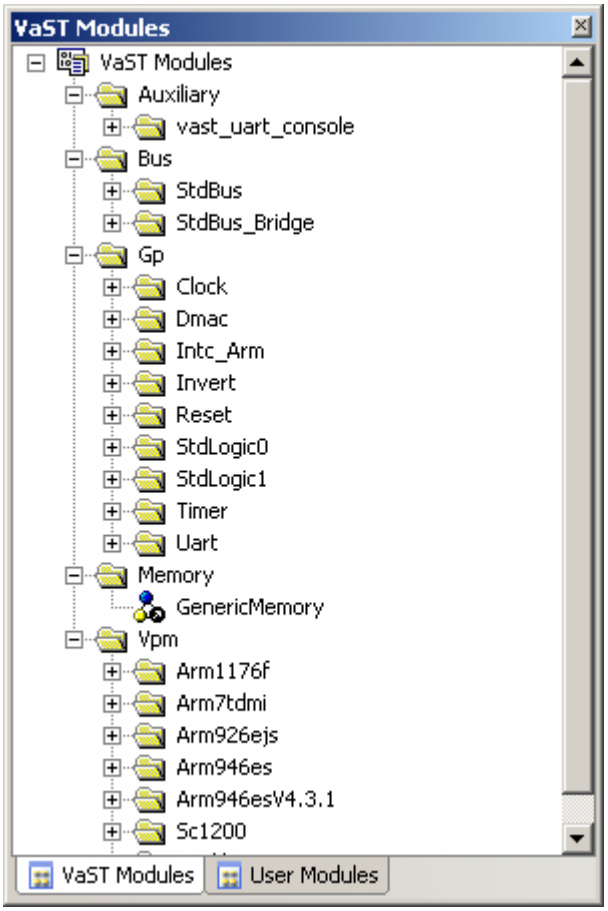
CoMET recursively searches the specified directory and its subdirectories, adding each module .dll it finds to the library. The library tree structure mirrors the directory structure CoMET encounters during its search. To determine the library tree structure of your library, arrange your module .dll files in the required directory structure.

You can specify a workspace or project directory as the library directory. CoMET finds any built .dlls and adds them to the library. This may result in redundant levels in the library tree structure.

On completion, CoMET adds a tab to the Modules window allowing navigation of the new module library.

To refresh the contents of the module window, choose **Tools/Rescan Modules**.

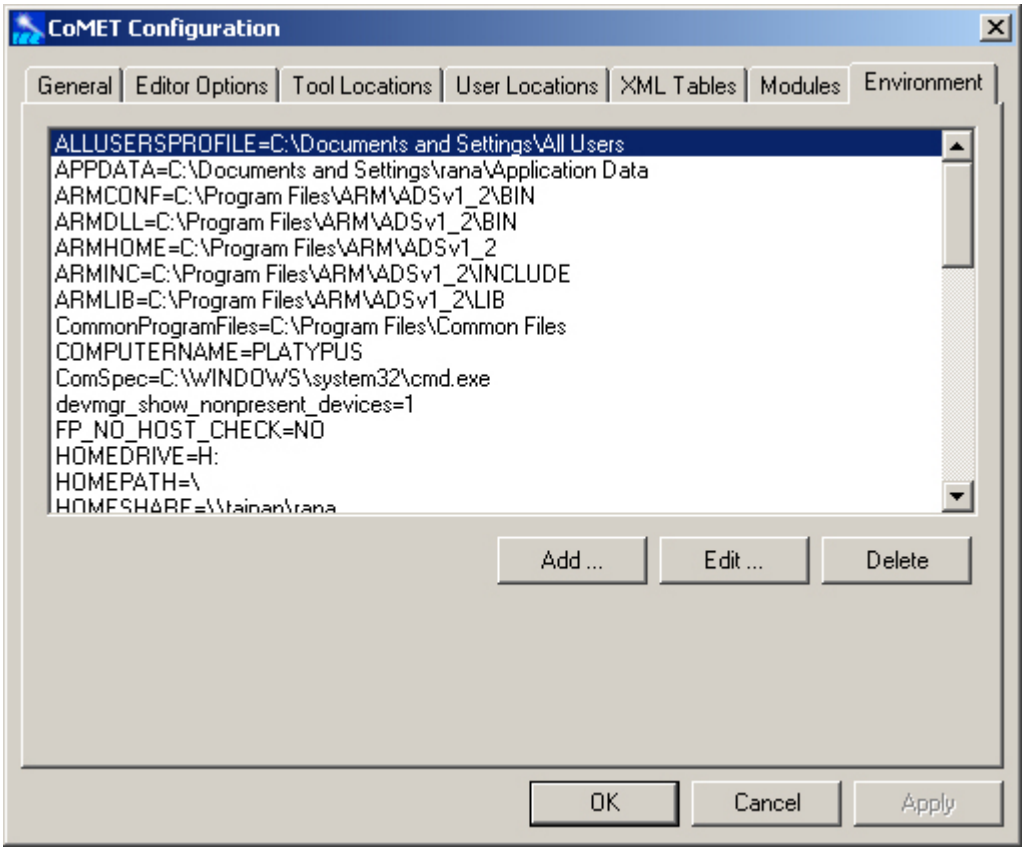
VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.



NOTE: CoMET must run each dll to extract the information required to build the library. If there is some issue preventing a dll running successfully, the library build process may not complete successfully.

Environment Configuration Settings

You can view the Windows system environment and add to or edit the environment variables.



Environment Configuration Tab

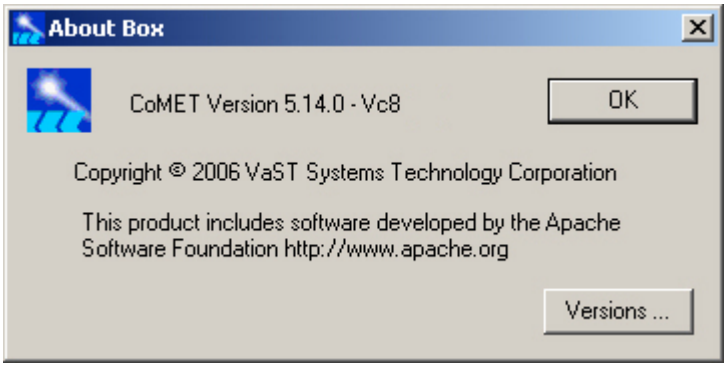
VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly
authorized in a written agreement with VaST.

Displaying CoMET and Tools / Models Versions

You can find out the version of the CoMET GUI or the versions of the tools and models via the **About CoMET** option from the **Help** menu.

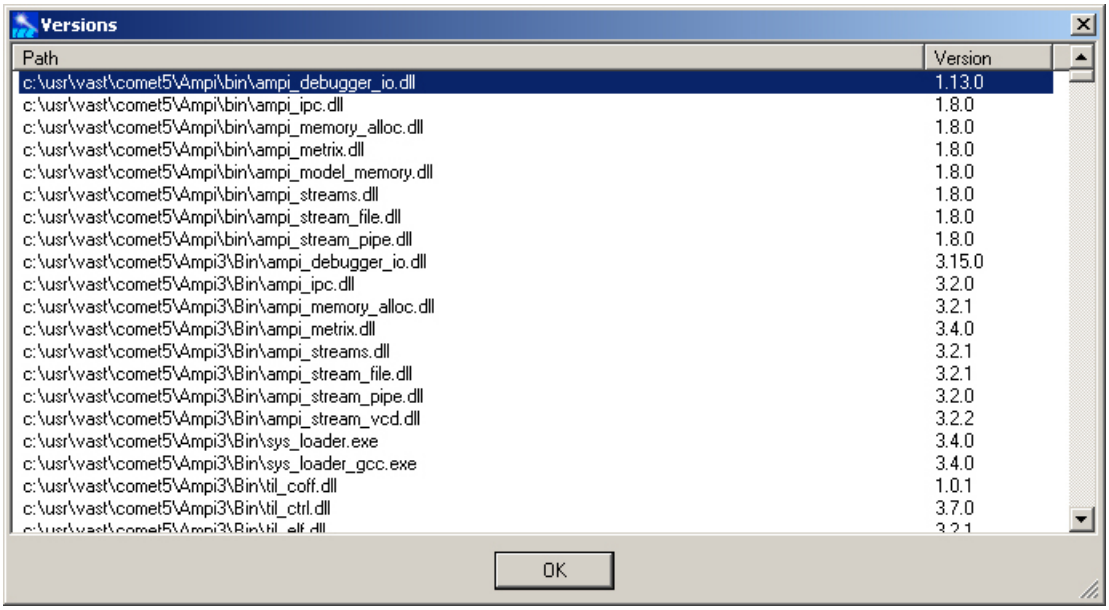
To display version information about the CoMET GUI:

- From the **Help** menu, select **About CoMET**
- To close the **About Box** dialog box click **OK**


About Box
CoMET Version 5.14.0 - Vc8
OK
Copyright © 2006 VaST Systems Technology Corporation
This product includes software developed by the Apache Software Foundation http://www.apache.org
Versions ...

To display version information about the tools and models within CoMET:

- Click the **Versions…** button in the **About Box** dialog box.



The list of tools and models can be sorted by **Path** or by **Version** number.

- Click **OK** to close the **Versions** dialog box.

Note: When you close the Versions dialog box using the OK button, the list of tools and models and their respective version numbers are automatically saved to the buffer. Thus to keep a record of the tools and models, simply save the buffer to a text file.

Getting Assistance while You Work

The Online Help contains extensive information on the CoMET System Engineering Environment. You can access Help by selecting any of the options from the **Help** menu, or the **Help** button on the tool bar.

For information on Help topics:

- From the **Help** menu, select **Help Topics**

or

- Press **F1**

CoMET Online Links

With Internet access, you can get more information about the CoMET System Engineering Environment on the World Wide Web.

CoMET Web Page

The VaST home page is located at:

http://www.vastsystems.com

CoMET Support Services

For all problems with installing or using CoMET, contact VaST by email at:

support@vastsystems.com

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Managing Workspaces and Projects

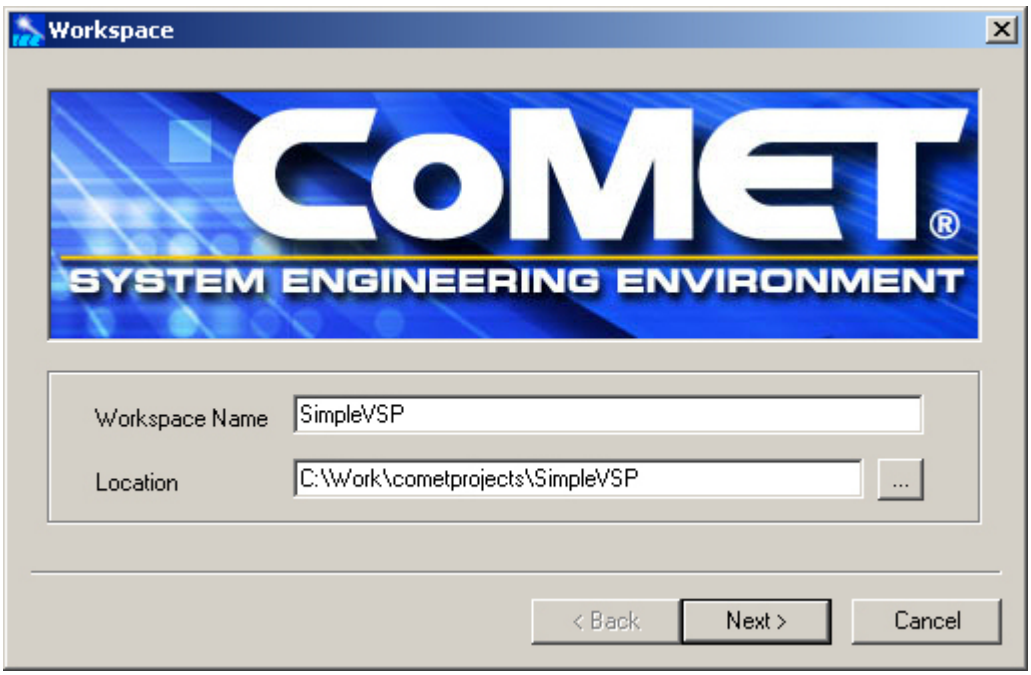
Each Virtual System Prototype you create consists of one or more modules. Modules are defined as projects (representing the physical device models) within the CoMET System Engineering Environment. A top-level module contains the structure of the Virtual System Prototype.

Creating a Workspace

The workspace contains the projects (that define the modules) in a Virtual System Prototype. A workspace may consist of a single project, or multiple projects.

To create a new workspace:

- From the **File** menu, select **New Workspace…**



NOTE: When you create a new workspace, CoMET automatically adds a new project to the workspace. To build a Virtual System Prototype, you must also add new modules (projects) or add existing modules (projects) to the workspace.

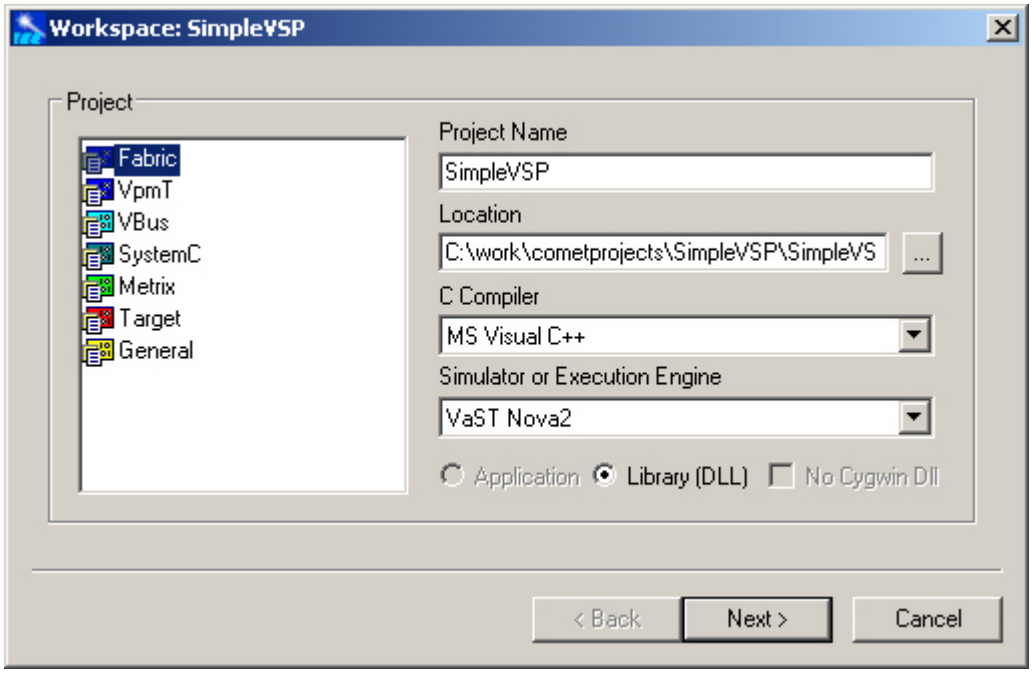
VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Adding a New Project

A project contains the configuration of a module (for example, the module files and their locations, settings for the compiler and simulator, etc).

To add a new project to the workspace:

- From the **Workspace** menu, select **Add New Project**



Project

A Virtual System Prototype can contain several types of project:

`Fabric` A project built with the VaST Communication and Infrastructure Fabric methodology. Fabric projects create modules that represent the physical device models within the Virtual System Prototype. For information about the VaST Communication and Infrastructure Fabric methodology refer to the *CoMET Modeling Guide using the Communication and Infrastructure Fabric*.

`VpmT` A VPM Transformer project. The VaST VPM Transformer is a toolset and methodology which allows users to modify the instruction set of a VaST Seed VPM. The VaST VPM Transformer Toolset needs to be purchased and installed before a VPM Transformer project can be created.

`Vbus` VBus project. For information on CoMET 5 support for VBus projects, see *Application Note: Converting CoMET 4 VBus Projects to CoMET 5 VBus Projects*

`SystemC` Standalone System C project. You can develop a pure SystemC project within the CoMET environment. A System C standalone model cannot be exported for use in METeor. A System C standalone model can later be interfaced with CIF models by migrating it to a Fabric module of module

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

type Peripheral Module SystemC. See *Defining a Fabric Module, page 47* and *Creating SystemC Projects, page 165*.

`Metrix` A project for tracing, profiling and dynamic debugging of your Virtual System Prototype. For information about Metrix, refer to the *Metrix User Guide.* An alternative to developing a Metrix project is to use a Metrix Configuration file and the generic Metrix dll. See the *Metrix Configuration File User Guide*.

`Target` A project that builds a target application executable. The target code runs on one or more VPMs defined within the Virtual System Prototype. CoMET currently supports the ADS ARM Tools and a number of GCC cross-compilers. Select the required compiler in the C Compiler field. For more details see *Application Note -- Third Party Tool Support*.

`General` A general-purpose project that you can use to build modules specific for your application, for example a test bench, analysis modules or other performance analysis tools.

Project Name

A name for the new project.

Location

The path of the folder in which the project files are stored.

The default location of a new project and the project files is the workspace directory. When the project name is specified, the name specified is appended to the location of the workspace, so the default project location becomes a sub-directory of the workspace. You can modify this path so that it points to another location by adding sub-directories, browsing to a different location, etc. Note that if you subsequently change the name of the project then this will change the text after the last directory separator to that of the specified project name.

C Compiler

The C Compiler used to build the source code files.

Simulator or Execution Engine

This selects the mode of simulation to be used for the module.

`Vast Nova 2` (Default): Select this for a CIF model. It is simulated with the Nova 2 Scheduler.

`Mentor Graphics ModelSim:` Select this to co-simulate your HDL device model with CIF using the ModelSim HDL simulator. See *ModelSim HDL Co-Simulation Support, page 169*.

`Vast Nova`: Select this for a (CoMET 4) VBus model. The model is built with a CIF wrapper, allowing it to be instantiated in a CIF platform. It is co-simulated with the Nova Execution Engine and the Nova 2 Scheduler.

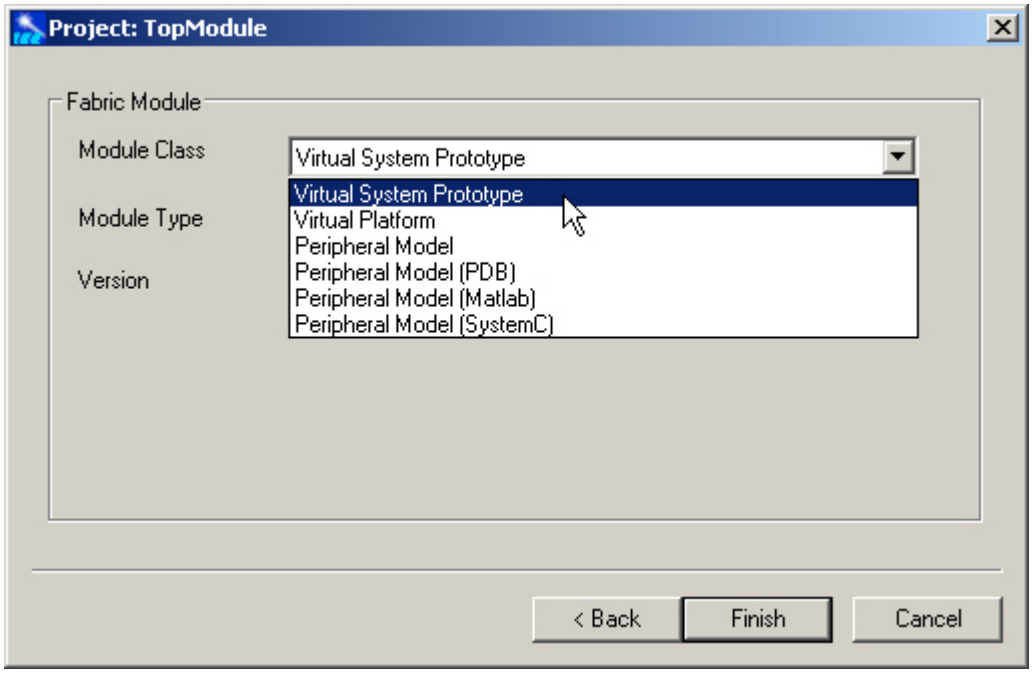
No Cygwin Dll Check Box

The No Cygwin Dll check box is selectable for Gcc projects. Check this box to automatically convert any Gcc project to use the no cygwin Dll option. See *Runtime Options: Auto-convert Gcc projects to No Cygwin Dll, page 28*.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Defining a Fabric Module

If you select a Fabric project, you must define the module class and the type of module.



Module Class

Each fabric module has a module class. Different module classes have different properties, such as build configuration and runtime options. For more details see *Setting the Build Configuration, page 109* and *Setting the Runtime Options, page 117*:

`Virtual System Prototype`

A module that defines the top of the VSP hierarchy. The top-level module differs from other modules in the hierarchy in that it does not contain any external connections.

`Virtual Platform`

A module that defines a hierarchy (instantiates other modules) but does not contain any behavioral functionality.

`Peripheral Model`

A module that contains behavioral functionality but does not instantiate other modules. In this case, a project is created that consists of a Peripheral Device Template with generic generated common functions. The template provides a reference upon which specific device models can be based.

`Peripheral Model (PDB)`

A module that is built and managed using the Peripheral Device Builder (PDB). See the *PDB User Guide* and associated documentation. This is available only to customers who have installed the PDB option.

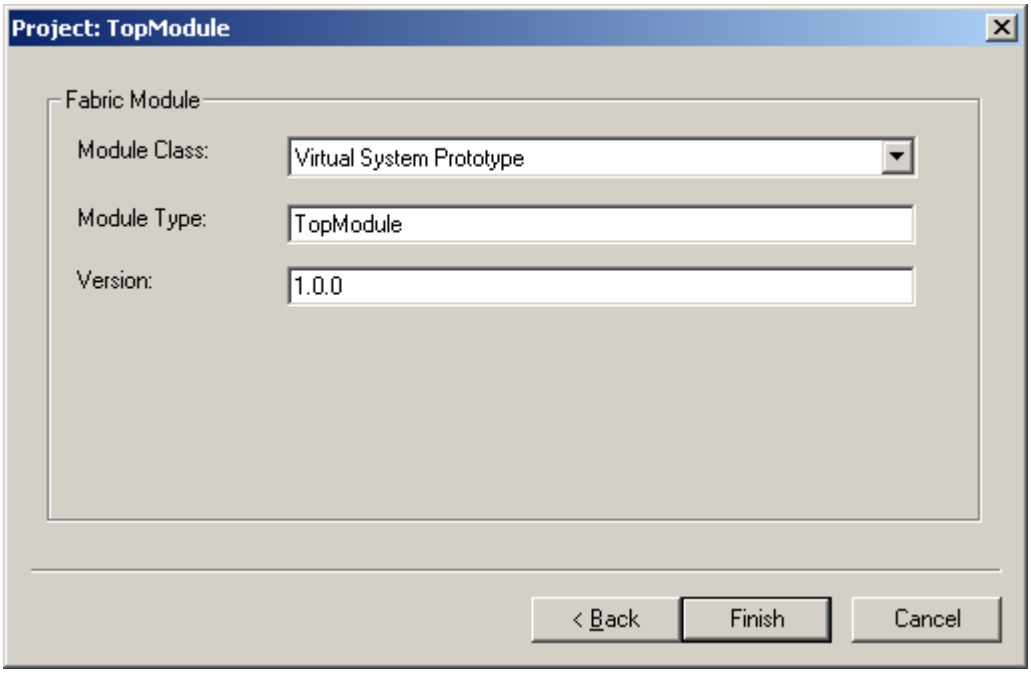
`Peripheral Model (Matlab)`

A module that interfaces a Matlab Model with a CIF Virtual Platform. This is available only to customers who have installed the Matlab Interface Device Generator (Matgen_CIF) option. See the *Matlab Interface Device Generator (Matgen_CIF) v1.0 User Guide*.

`Peripheral Model (SystemC)`

A module that interfaces a SystemC module or modules with a Fabric platform. The project skeleton code for a SystemC adapter module with interfaces to both a Fabric bus and a sample SystemC device module.

See *Creating SystemC Projects, page 165*.



Module Type

A Module Type name. The name is system-generated, and is the type of module that the project will represent. Separate instantiations of this Module share this Module Type. You can change the predefined value to any desired name. Note that you cannot use spaces in the name.

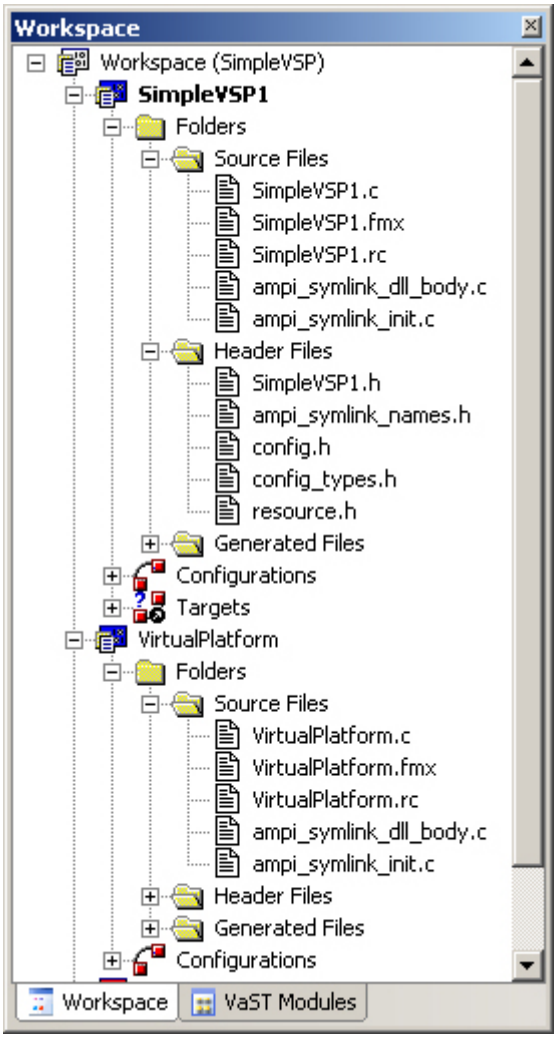
VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Version

A version number for the module. You can use the version number for version control of modules. For example, you can build several versions of the same module for use in different VSPs, or use the latest version of the module in all VSPs.

Project Configuration

When you add a new project to the workspace, a set of source files (arranged in folders), configurations and targets are included in the project.

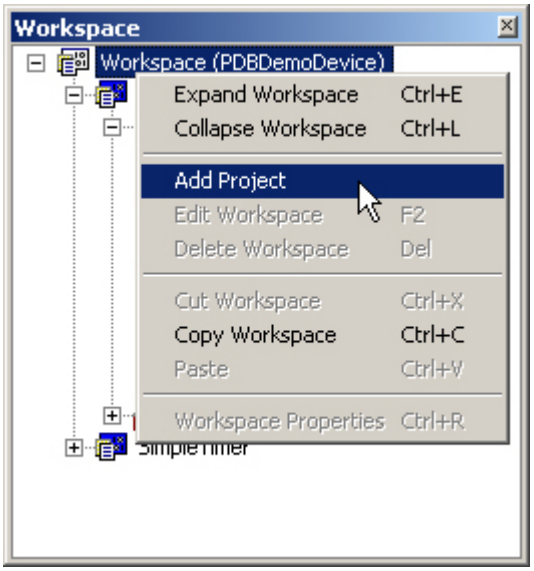


Adding an Existing Project

You can add existing projects to the workspace.

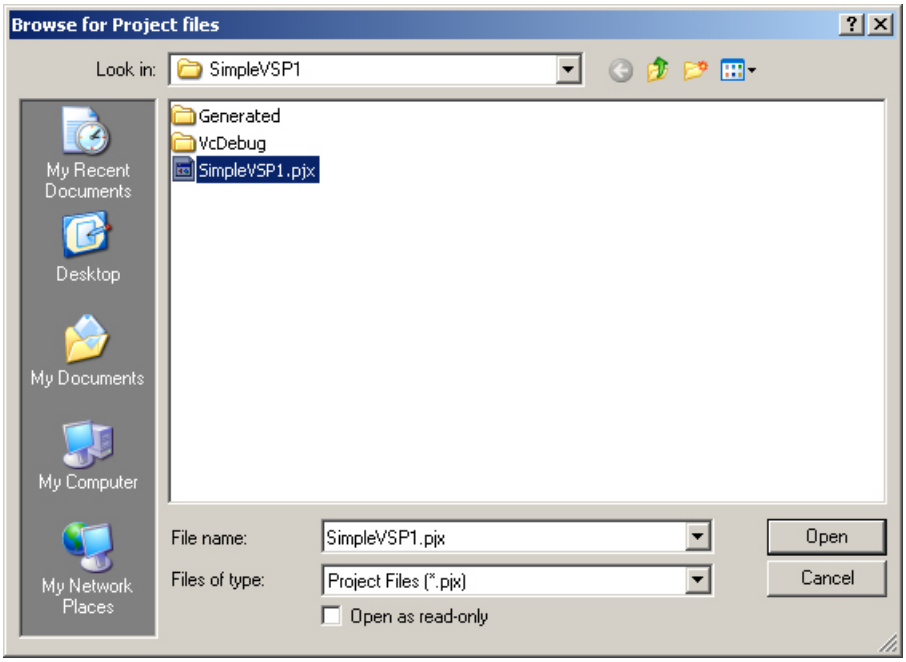
To add an existing project to the workspace:

- Right-mouse click on the workspace icon and select **Add Project**



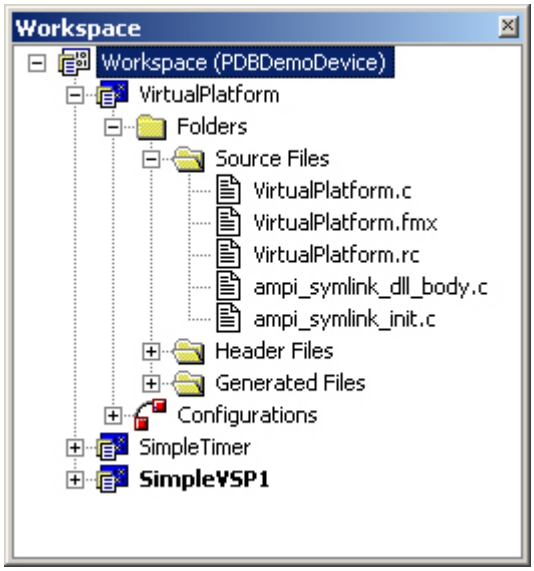
or

- From the Workspace menu, select Add Existing Project



- Navigate to the location of the project, select the .pjx file and click the **Open** button.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.



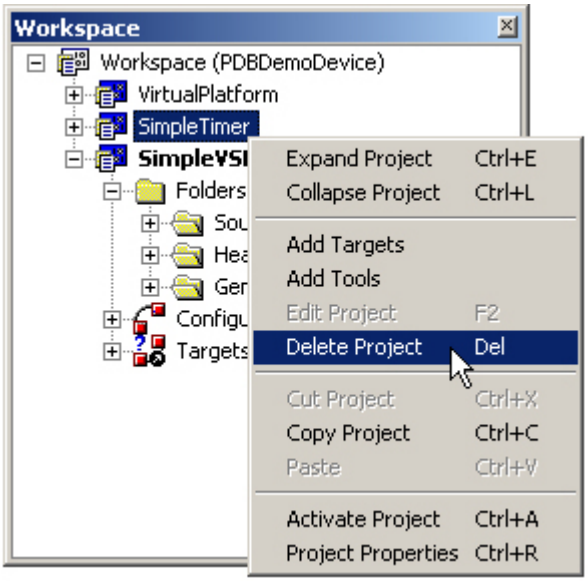
Removing a Project from the Workspace

To remove an existing project from the workspace:

- From the Workspace menu, select Remove Project

or

- Right-mouse click on the project icon and select **Delete Project**



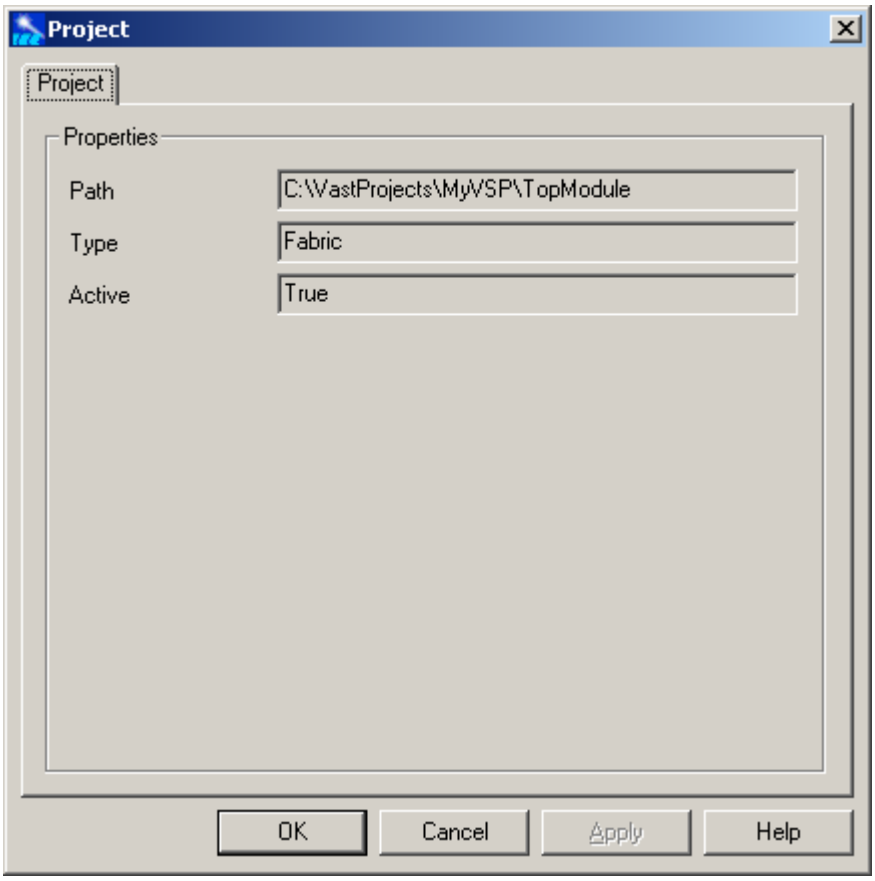
Viewing the Properties of a Project

To view the properties of a project:

- Double-click on the project icon

or

- Right-mouse click on the project icon and select **Project Properties**



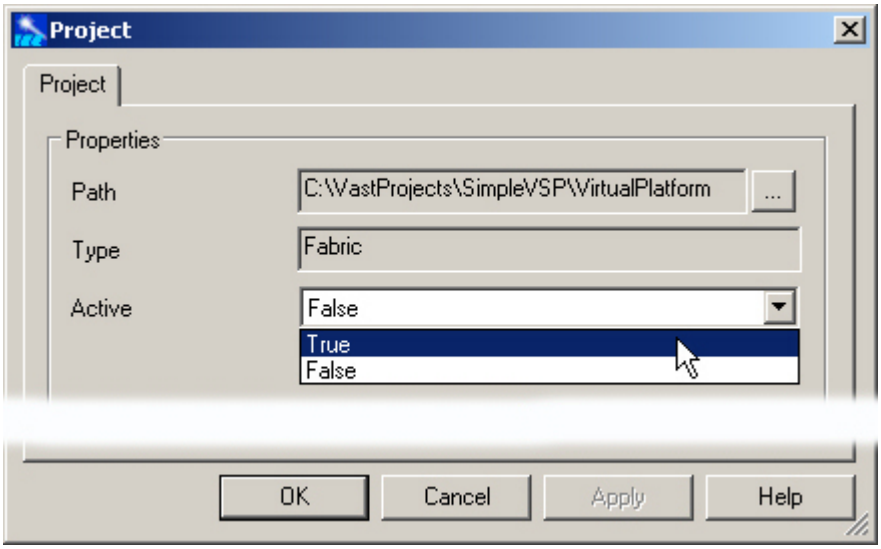
VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly
authorized in a written agreement with VaST.

Setting the Active Project

A CoMET workspace may contain several projects. When you build the workspace and execute the Virtual System Prototype, the active project is built and executed.

To set the active project:

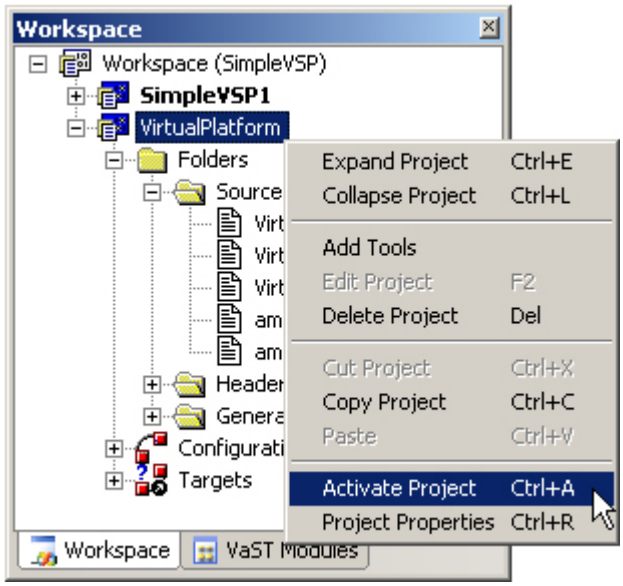
- Double-click on the project icon



- Set the Active field to True and click the **OK** button.

or

- Right click the project in the **Workspace Window** and choose **Active Project** from the context menu.



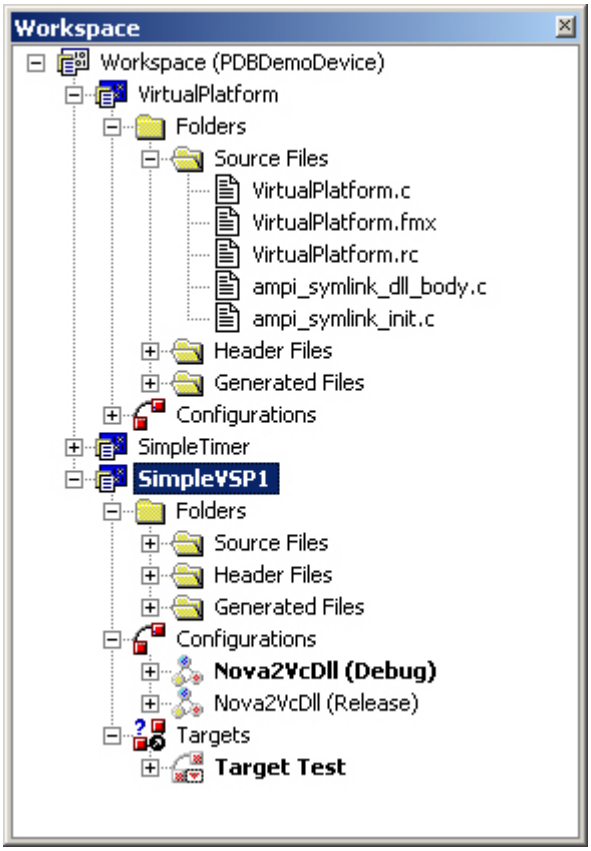
or

- Select the project in the Workspace Window and press Ctrl-A.

Configuring a Project

When you create a new project, depending on the project type, CoMET automatically generates a set of files and build configuration options. Generated files contain skeleton code, providing a basis for further development.

This list of files and options is maintained in the Project (.pjx) file. You can add and remove components such as source files and other alter options via the Workspace window interface. This interface follows the general CoMET methodology for managing configuration files. Right click on an element and select an action from the context menu.



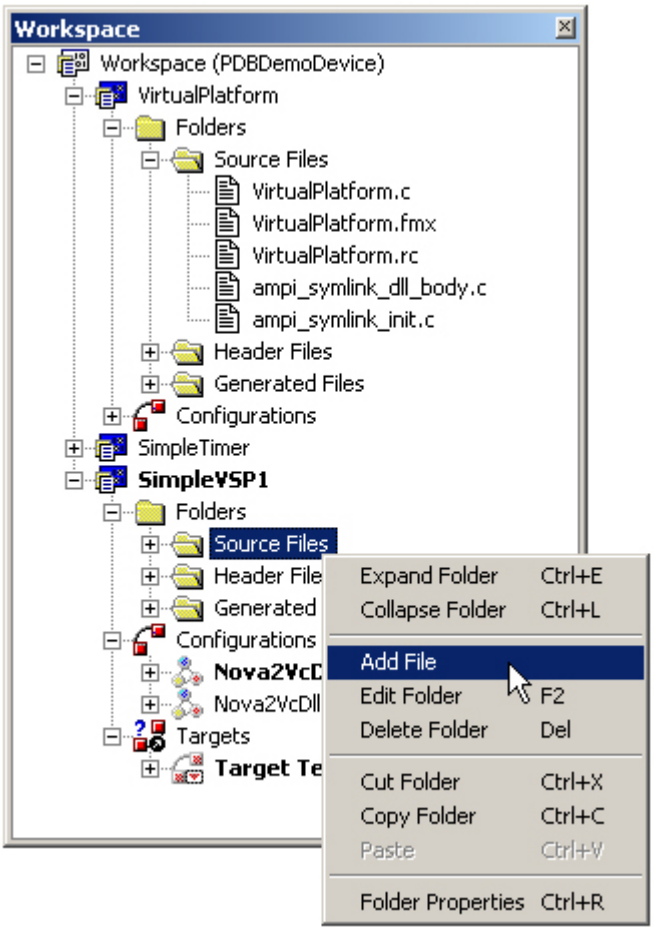
To set or change project components:

- Right-mouse click on the appropriate component icon and select the appropriate command.

For an example, see *Adding a Source File to a Project* below.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Adding a Source File to a Project



To add a source file:

- Right click on a Folder under the Folders element
- Choose **Add File** from the context menu
- In the **Browse for Source Files** dialog, locate the file to add.
- Click **Open** to add the file to the project.

Viewing Project Component Properties

To view project component properties:

- Right-mouse click on the project component and from the context menu select the corresponding **Properties** choice, e.g. **Folder Properties**.

Properties of Special Components

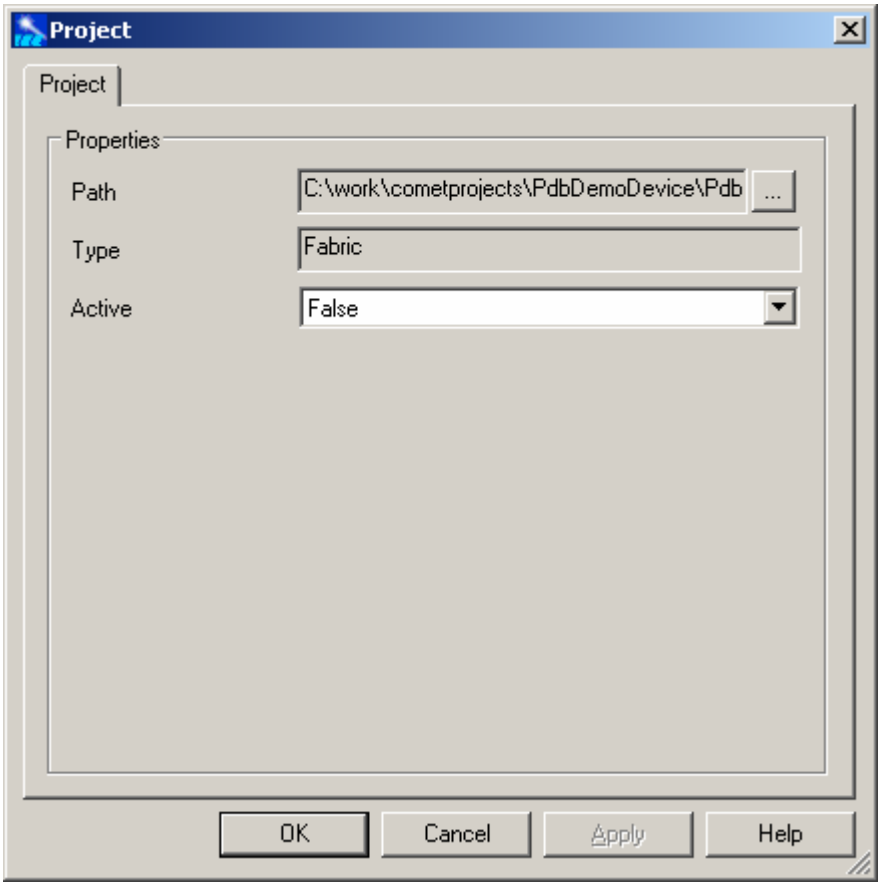
Projects, Configurations, RunTime Options and other components have a number of properties. These are displayed for viewing and editing in a Properties dialog.

Viewing the File Path Property

File components have a single property, the path of the file. CoMET displays the path property for viewing and editing in a browser window rather than a properties dialog.

Disabled Properties

Some components, such as the **Folders** Folder, do not permit viewing and editing of properties, so the Properties context menu item is disabled.



Workspace element properties dialog example: Project properties

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Saving Workspace Files

To save the Workspace and all files open in the Document Window:

- Choose **File/Save Workspace**.

To save all files open in the Document Window:

- Choose **File/Save All Files**.

Exporting a Virtual System Prototype Workspace for METeor

Virtual System Prototypes developed in CoMET can be exported as a Workspace to a file, for import by METeor. METeor users can develop target software and Metrix projects and perform simulations on the VSP.

VBus standalone projects are exported as a Workspace containing a single VSP Project. CIF Workspaces can be exported with multiple projects, including CIF projects, Target projects General projects and Metrix projects.

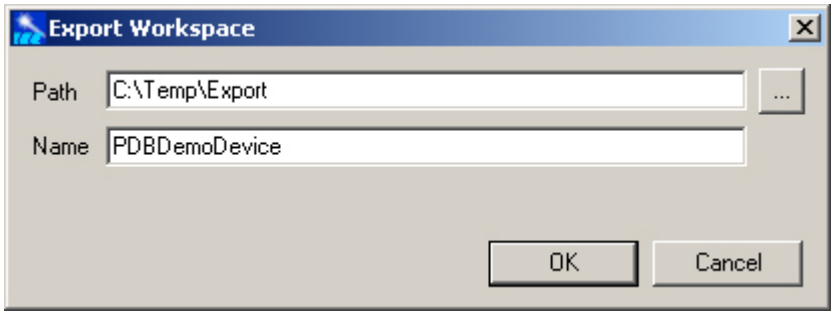
NOTE: To successfully export a CIF workspace, all CIF projects in the workspace must be of the same build type and configuration. All projects must be in either Debug or Release configuration, but not a mixture of both. All projects must use the same compiler, either Vc or Gcc, but not a mixture of both.

To export a VSP, you export the Workspace containing the VSP.

Exporting a Workspace

To export a Workspace:

- Choose File/Export Workspace

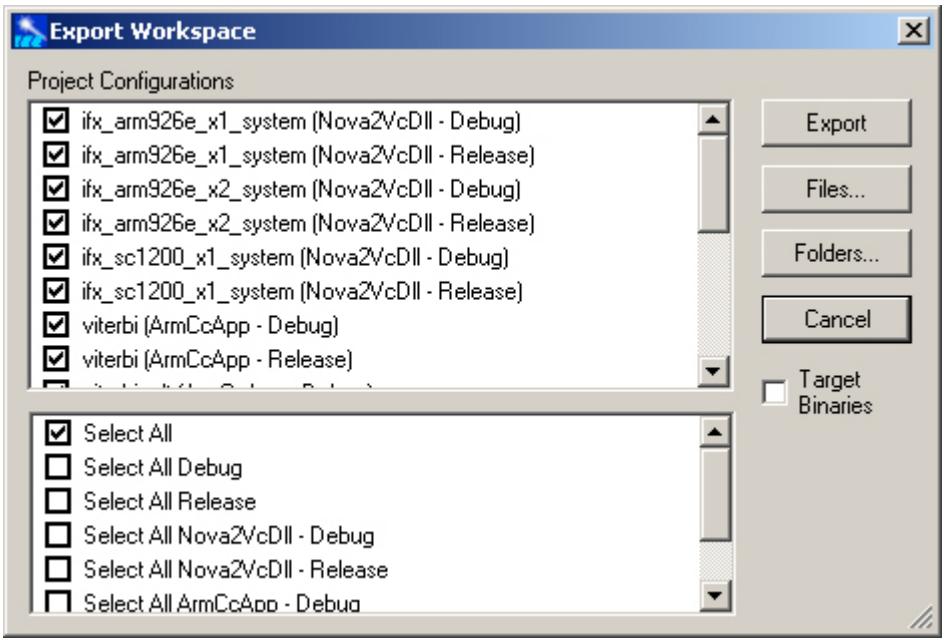


Export Workspace dialog:

- **Path**: Enter the required path for the exported file
- **Name**: Specify the required name for the exported file
- Click **OK**

CoMET displays the Export Workspace dialog Project Configurations list.

Choosing Export Options



In this list you can specify one or more configurations to export. What files are exported depends on the type of project you select.

Files Exported for Specified Projects

Fabric projects: CoMET exports the .dll files for all modules within the platform. It also exports the project (.pjx) file and the Prototype Configuration File (.pcx) associated with the selected Build Configuration.

You can choose to export the target binaries associated with the project. To export associated target binaries, check the **Export Target Binaries** check box.

Target projects: CoMET exports all source files and supporting files for the project. Using these you can rebuild the target binary in METeor and associate it with the METeor project.

Metrix projects: CoMET exports all source files and supporting files for the project.

Ensuring the Fabric project Prototype Configuration (.pcx) file is exported

The .pcx file associated with the build configuration can be viewed and modified in the Runtime Options for that configuration. See *Setting the Runtime Options, page 117*. The .pcx file is automatically added to the Runtime properties when it is created. When you build a project, CoMET checks for an existing .pcx file and if none exists, CoMET prompts you to create one. If you can successfully simulate your project, you have a valid .pcx file that will be exported with the Fabric Workspace.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Exporting Additional Files and Folders

You may wish to export additional support files, such as configuration, data, or documentation files. You can add individual files or entire directories for export with the project. You can add individual files to be included in the exported Workspace.

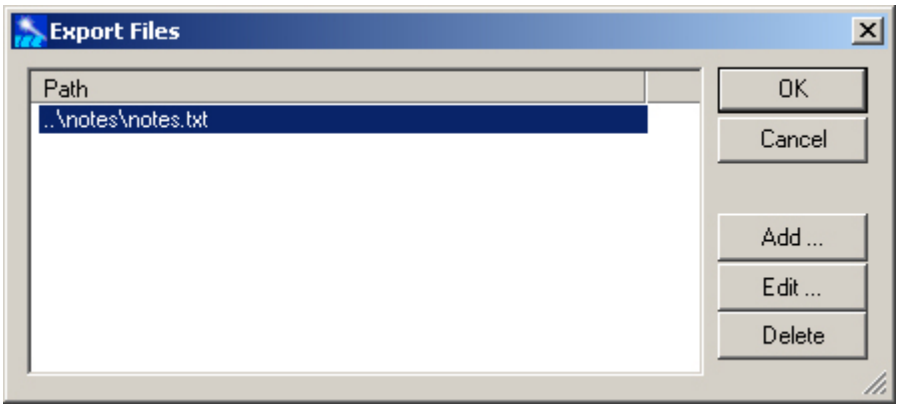
In the **Export Workspace dialog**, click on the **Files…** button. CoMET displays the **Export Files** dialog.

To add a file, click Add, then browse to locate the desired file.
Repeat until you have specified all desired files.

To remove a file from the list, select the file and click Delete.

To directly edit the file path, select the file and click Edit.

When the list is complete, click OK.



You can add an entire folder to be included in the exported Workspace.

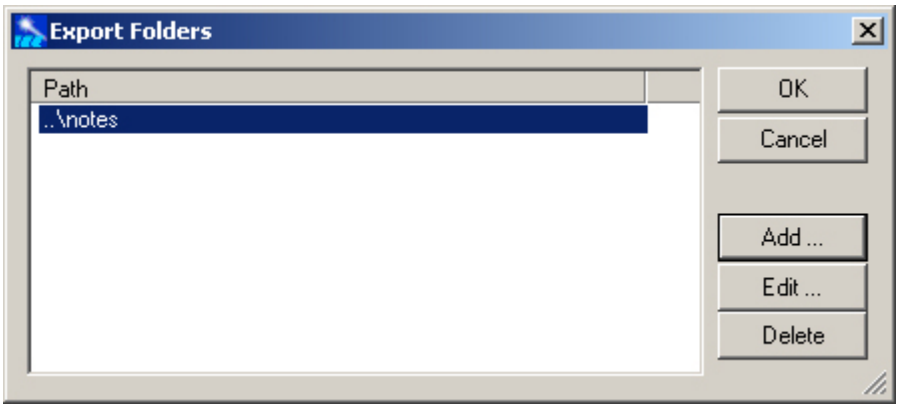
In the **Export Workspace** dialog, click on the **Folders…** button. CoMET displays the **Export Folders** dialog.

To add a folder, click Add, then browse to locate the desired folder.
Repeat until you have specified all desired folders.

To remove a folder from the list, select the folder and click Delete.

To directly edit the folder path, select the folder and click Edit.

When the list is complete, click OK.



Location of extra files and folders in Workspace

CoMET determines the location of extra files and folders by their path relative to the Workspace folder. If extra files or folders are within the Workspace folder, they appear in the same relative path within the exported folder.

If an extra file is outside the workspace, it appears in the exported workspace under the directory Extra, under a directory with the same name as the file's original parent directory.

For example, the file `c:\work\notes\notes.txt` appears in the exported workspace as `\extra\notes\notes.txt`

If an extra folder is outside the workspace, it appears in the exported workspace directly under the directory Extra.

For example, the directory `c:\work\notes\` appears in the exported workspace as `\extra\notes`

Creating the Exported Workspace .cab File

When the export options are complete, click the **Export** button to export the Workspace. CoMET creates a file with the specified path and name and a .cab extension.

To import into METeor, choose **File/Import Workspace** and browse for the .cab file containing your exported workspace.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Moving a workspace from one directory to another

When you move a workspace from one directory to another, you may need to reestablish the paths of files within the project, depending on the original location of the files.

The CoMET workspace (.wkx) file records paths of the project (.pjx) files belonging to the workspace. The Project (.pjx) file records the path of component files and directories, such as includes or libraries.

Workspaces are represented on disk as directory trees. The .wkx and .pjx files record all paths as relative, except for paths residing in the CoMET home directory (determined by the environment variable `%VAST5_HOME%`). These are recorded as absolute with `%VAST5_HOME%` as root.

If you move the workspace directory, the project file paths remain valid for project files residing under the workspace directory. Similarly, the project component paths remain valid for components residing under the workspace root directory.

Paths to files or directories residing in the `%VAST5_HOME%` are automatically reestablished after moving a workspace to another directory.

However, if a .wkx file or .pjx file refers to a file or directory outside both the workspace directory and the `%VAST5_HOME%` directory, the file path becomes invalid if you move the workspace directory.

In this case, you must reestablish the file path manually.

To reestablish project and component file paths:

- open the project in the CoMET Workspace Window
- for each project component browse to find the correct path

Alternatively:
Edit the workspace (.wkx) and project (.pjx) files directly in a text or xml editor.

May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Developing a Fabric Module Definition

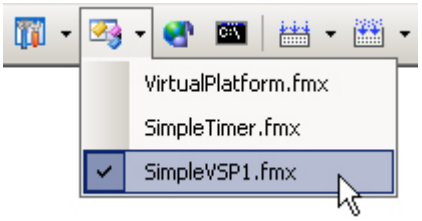
The module structure defines the characteristics of the module, and how the module interconnects with other modules within the Virtual System Prototype. Module structure and properties are defined in the Fabric Module definition (.fmx) file for the module.

By editing the .fmx file you can add ports, nets, module instances, tasks, parameters and other properties to the module.

Opening the Fabric Module Definition file

To open the Fabric Module Definition file:

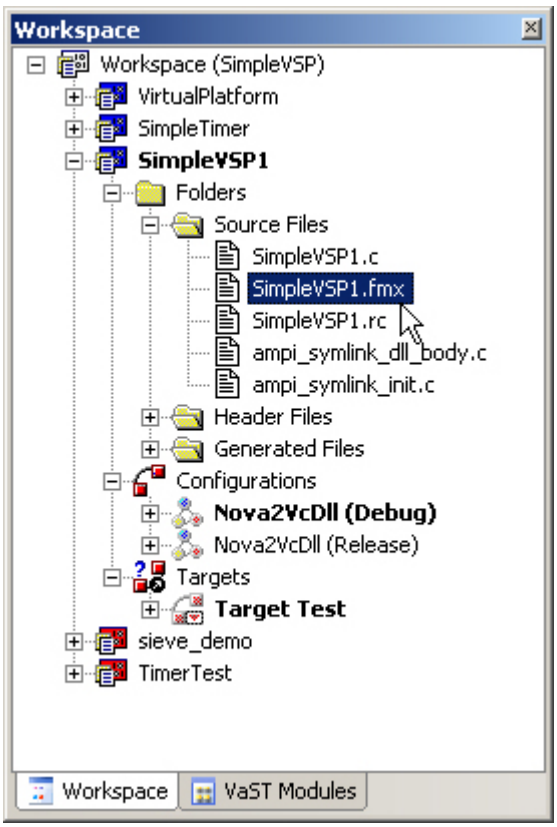
- Select the module from the **Fabric File** button



or

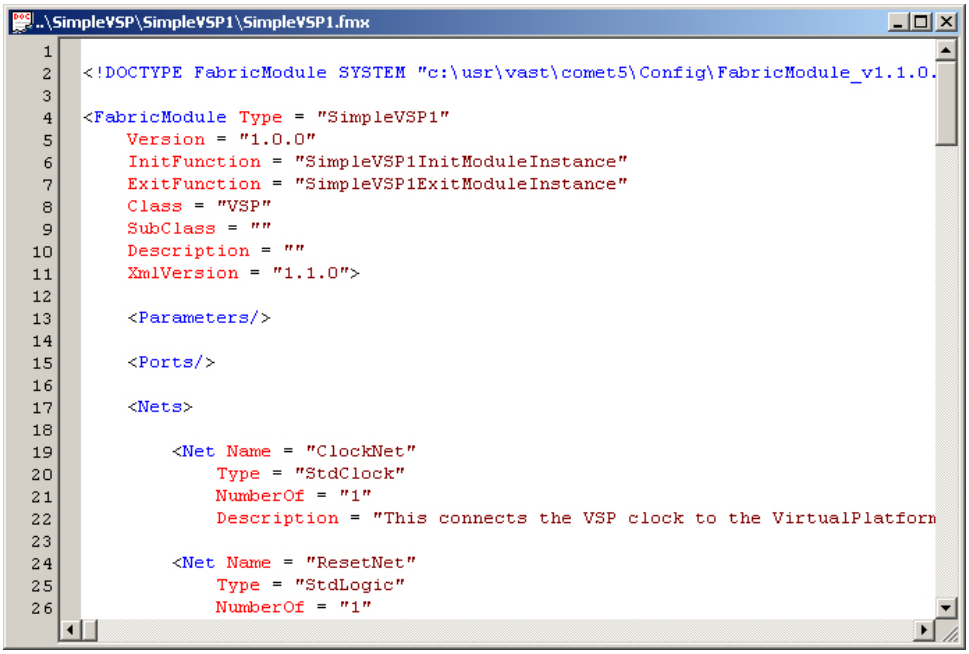
- Expand the module node (in the workspace window) and navigate to the fabric module definition file (<*module name*>.fmx).

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.



- Double-click on the Fabric Module Definition file (.fmx).

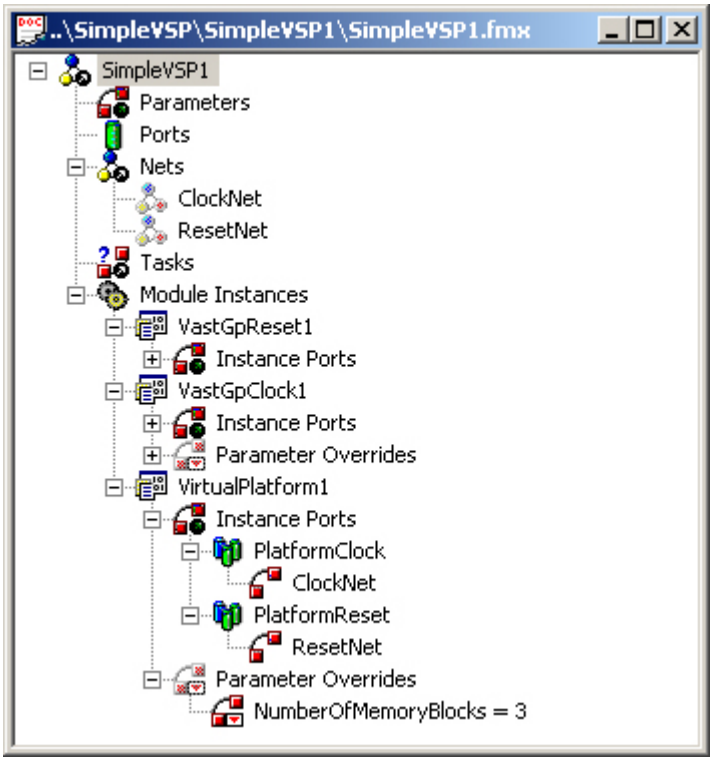
The .fmx file opens in a document window. Several views of the .fmx file are available:



XML Text View

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly
authorized in a written agreement with VaST.

Right click on the Document Window to select the XML Tree View from the Context Menu.



XML Tree View

There are also various table views: Memory Parameter, Bus Connection, and so on. See XML Tables Configuration Settings, *page33*.

Defining the Fabric Module involves editing the .fmx file to add and specify the details of elements such as Parameters, Ports, Nets, Tasks, Module Instances and Interfaces.

Each change modifies the .fmx file. Depending on the changes you make, CoMET may make further modifications to the .fmx file, and may generate changes in other files.

Editing the Fabric Module Definition file

The CoMET Methodology involves first creating the Fabric Module Definition, then creating any behavioral code required.

Use the Project wizard to create a CIF project. Choose Workspace/Add New Project and choose a Fabric project type. This ensures that the appropriate templates are used for source and fmx files.

Creating the Fabric Module Definition involves editing the Fabric Module Definition (.fmx) file. This can be done by directly editing the XML file in XML Text View, but it is recommended to use the XML Tree View interface. XML Tree View provides validation and other safeguards to ensure a valid .fmx file is generated.

For each element in the specification there is a node in the XML tree, and a corresponding properties dialog.

The remainder of this chapter provides details for each of the elements and its properties dialog. Each dialog corresponds to an element, such as a Port or Net. For each dialog and corresponding element, attributes are listed with a full description. Dialogs are listed according to their path from the top of the Fabric Module tree.

For example, the Port element dialog, which appears when you add a Port to the Ports node of the Fabric Module, has the path:

Fabric Module / Ports / Port

A double slash in the path indicates there may be intervening elements. For example:

Fabric Module / Parameters // Parameter

This path implies that the Parameter element must have an ancestor Parameters element, but may have closer ancestors. In the case of the Parameter element, closer ancestors may be SubParameters elements.

Prerequisite Elements

The path of an element indicates some prerequisites to creating the element. For example, before creating a

Fabric Module / Task / Port Or Net Views / Port Or Net View

element, you must first create a

Fabric Module / Task / Port Or Net Views

element.

Some elements may require, as prerequisites, elements in other branches of the tree. For example, if you wish to define the value of a Port attribute as a Parameter, you must first create the Parameter. To make a **Module Instances / Module Instance / Port Connections / Port Connection**, you must first create the Port to which it connects. Elements are arranged in the XML Tree View in an order indicating which should be created first. It is usually more convenient to first create elements which appear earlier in the hierarchy element order. However the methodology is flexible, and you can create elements in any order.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly
authorized in a written agreement with VaST.

Choosing Names

Most elements have a name attribute, which, together with the position in the Fabric Module Definition hierarchy, uniquely identifies the element. Choose names carefully. Names appear in generated code, views and reports. A name should therefore not contain characters disallowed in C code. Other useful features of a name include being easy to recognize, and carrying some useful meaning, such as function or type.

A strict naming convention, appropriate to your project, is recommended.

Names may also include an array declaration, or a reference to array members. See *Arrays*, below.

Description attributes

Many elements contain a Description attribute. Use these to provide documentation for your Fabric Module and its elements. Descriptions appear in Prototype Reports. See *Viewing Module Definition and Prototype Configuration Reports, page 100*.

Arrays

You can declare **Port**, **Net** and **Module Instance** elements as arrays. This allows a single element definition to take the place of multiple similar elements, and simplifies operations on multiple similar elements. Members of arrays share attributes.

Array notation provides for declaring element arrays and referring to members within the array.

See *Port, Net* and *Module Instance* below.

Declaring an Array

Element arrays must be declared with the required number of members. To declare an element array you can use either the **Number Of** attribute or an array notation declaration in the element name.

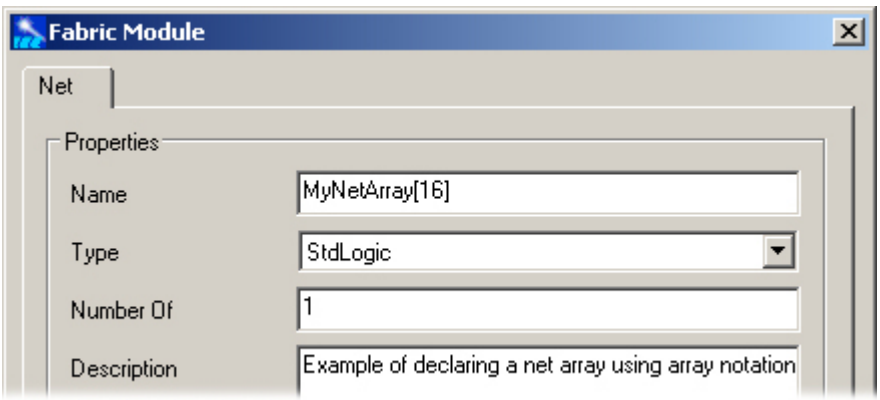
Array Notation Declaration

You can declare an array of elements by using array notation in the element name. An array notation declaration follows the element name with square brackets containing the dimension of the array, using the same syntax as declaring an array in C.

For example, to create a 16 element array, specify the name of a Net as:

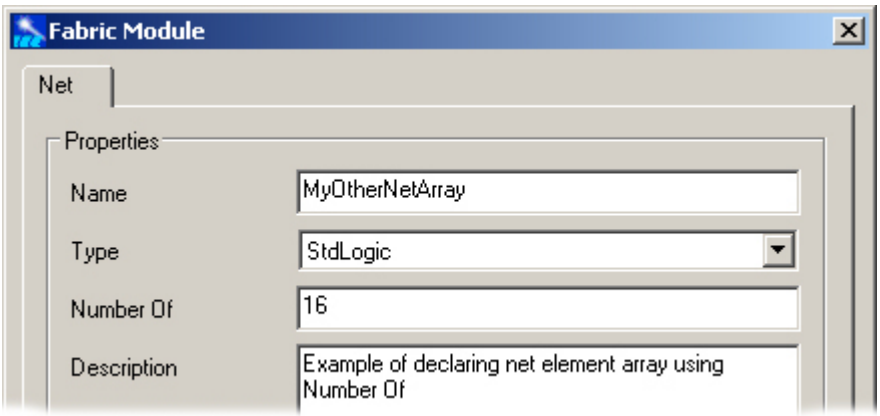
```
MyNetArray[16]
```


May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.



The Number Of Attribute

An element that may be defined as an array has a Number Of attribute that defines the dimension of the array. The default value for Number Of is 1.



VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Referring to Members of an Element Array

You need to specify array members when referring to arrays of nets, ports or module instance ports. You may specify array members for the following elements within an .fmx file:

- A Module Instance Port name
- A Port Connection Port or Net name
- A Port Or Net View name

Array notation provides a compact and efficient way to referring to groups of array members.

When creating a connection, between members of an Instance Port array and members of a Port or Net array, array notation allows you to name the element to specify:

- individual array members (by index): *ElementName*[i]
- a list of array members (comma separated indexes}: *ElementName*[i,j,k]
- a range of members (start index : end index }: *ElementName*[i:k]
- a combination of list and range: *ElementName*[i:k, j]

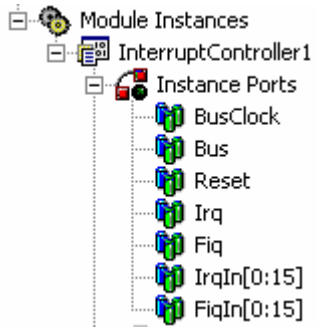
Port or Net View elements may also refer to members of Port or Net arrays, using array notation. See *Port Or Net View*, *Instance Port* and *Port Connection* below.

Using Array Notation in an Instance Port Name

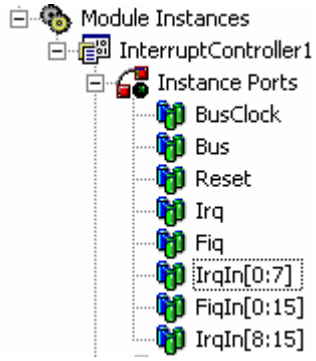
When you add a Module Instance to a platform, CoMET creates a single Instance Port for each Module Port. You can create subgroups of Instance Port array members by adding additional Ports and naming them using array notation.

For example, the Instance Port name IrqIn[0:15] refers to members 0 to 15 of the IrqIn element array.

You can rename this Instance Port name to IrqIn[0:7], and add another Instance Port named Irq[8:15], to permit you to connect the two groups of array members independently.



Default module instance with a single Instance Port for each Module Port



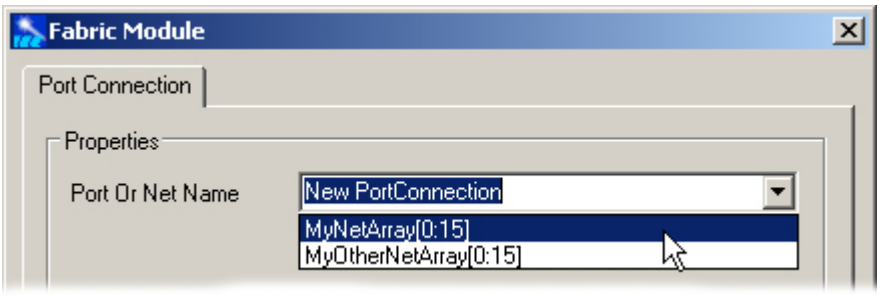
Modified module instance, with an additional Instance Port, allowing two groups of IrqIn array members to be connected independently

Using Array Notation in a Port Connection Port Or Net Name

CoMET distinguishes a Net or Port array by listing it with an element range in the drop down list for the Port Or Net Name field. You can edit the range to specify the required elements.

When you specify an Instance Port/Port Connection:Port Or Net Name, you can use array notation to indicate a selection of array members. CoMET connects the members of the expanded Instance Port array members list to the respective members of the expanded Port Or Net Name list. The number of members in each list must match.

For example, to connect the first group in the above Instance Port example, add a Port Connection to the Instance Port IrqIn[0:7]. You can now specify a connection to 8 members of a Net array.



To connect member Irq[0] to member MyNetArray[0], Irq[1] to MyNetArray[1]... and Irq[7] to MyNetArray[7] specify the Port or Net Name as:

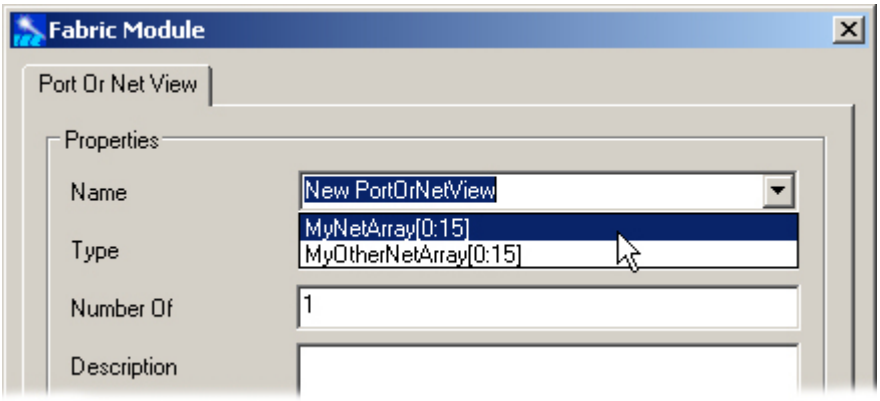
```
MyNetArray[0:7]
```

To connect member Irq[0] to member MyNetArray[7], Irq[1] to MyNetArray[6]... and Irq[7] to MyNetArray[0] specify the Port or Net Name as:

```
MyNetArray[7:0]
```

Using Array Notation in a Port or Net View

CoMET distinguishes a Net or Port array by listing it with an element range in the drop down list for the Port Or Net View:Name field. You can edit the range to specify the required members, or specify the members in the Number Of attribute, omitting brackets.



VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Options for Referring to Array Members

The table below shows the various array notation options for specifying elements.

In the Example column the notation ⇔ indicates a connection between the Instance Port array members on the left and the Port or Net array members on the right.

Connection Type	Instance Port(s)	Port or Net connection(s)	Example
Signal to signal	SignalA	SignalB	ResetIn ⇔ ResetModule
Bit to bit	SignalA[x]	SignalB[y]	IrqIn[3] ⇔ IrqModule[1]
Bit range to bit range	SignalA[0:x]	SignalB[2:y]	IrqIn[4:7] ⇔ IrqModule[0:3]
Bit range to bit range (twisted)	SignalA[0:x]	SignalB[x:0]	IrqIn[4:7] ⇔ IrqModule[3:0]
Bit list to bit list	SignalA[x1,x2,x3]	SignalB[y1,y2,y3]	IrqIn[1,3,5,6] ⇔ IrqModule[0,2,3,4]
Bit list and/or range to bit list and/or range	SignalA[x1,x2:x3,x4]	SignalB[y1:y2,y3]	IrqIn[0,2,4:7] ⇔ IrqModule[0,2:6]
Single bit connection to bit list and/or range	SignalA[0:x]	SignalB[x] or SignalB	IrqIn[15:31] ⇔ SomeSignal
Bit list or range connection to single bit	SignalA or SignalA[x]	SignalB[y1:y2,y3]	SomeSignal ⇔ IrqIn[15:31]

Fmx Update - Update Fabric

An Fmx file may contain a module instance of a module still under development. When the module changes, the derived module instance must change to reflect those changes. In particular, the Instance Ports of the module instance must correspond to the Ports in the changed module, and the Parameter Overrides of the module instance must correspond to the Parameters in the changed module.

The Fmx Update process accomplishes this. Fmx Update occurs as part of the Build process and may also be initiated by choosing a menu option. To update the active project fmx file, choose **Workspace/Update Fabric**. CoMET determines all modules that contribute to module instances within the active fmx file, and working from the lowest level of the hierarchy, updates module instances up to the active fmx file.

Instance Port/Port Connections and Parameter Override values in a module instance are preserved by the Fmx Update process where the Port or Parameter in the module is unchanged. New module Ports and Parameters result in new Instance Ports and Parameter Overrides in the module instance. Where a Port or Parameter is deleted in the module, the corresponding Instance Port or Parameter Override is deleted from the module instance, and connections and values are lost.

See *Port, Parameter, Module Instance, Parameter Override* and *Instance Port* below.

Prototype Configuration Files use the same mechanism to accommodate changes to module definitions. See *Pcx Update, 122*.

Child Elements

Some elements may contain child elements. For example, a SubParameters element may contain one or more Parameters and one or more SubParameters elements. In the element descriptions below, where an element may have child elements, the child elements are listed. The top Fabric Module element is created with some automatically generated child elements. Depending on the project class, these may include:

- Parameters
- Ports
- Nets
- Tasks
- Module Instances
- Interfaces

Creating a Child Element

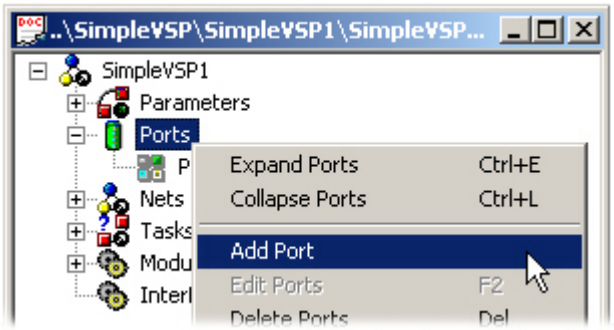
To create a child element:

- Open the Fabric Module Definition (.fmx) in the Document window XML Tree View
- Right click on the node corresponding to the parent element
- From the context menu, choose the Add action for the child element you wish to create

For example, to create a Port, right-click on the Ports element and choose **Add Port** from the context menu. The dialog for the **Fabric Module / Ports / Port** element appears.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly
authorized in a written agreement with VaST.

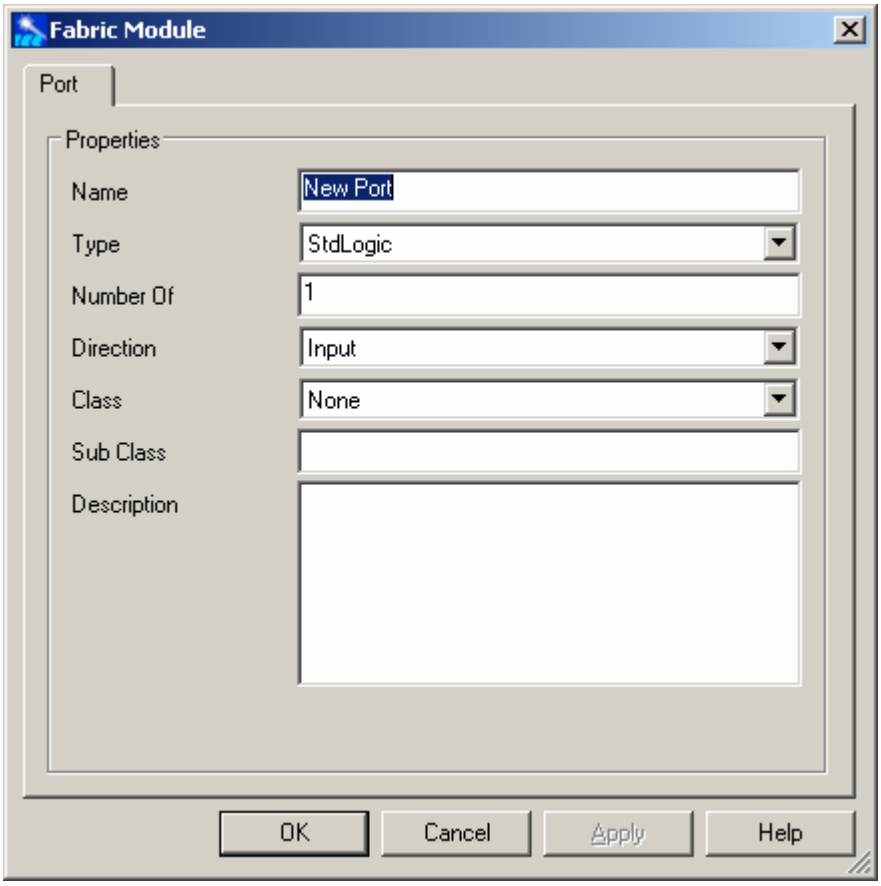
An Add action is available on the context menu only if the element prerequisites exist.



CoMET opens the dialog corresponding to the element.

The Element Dialog

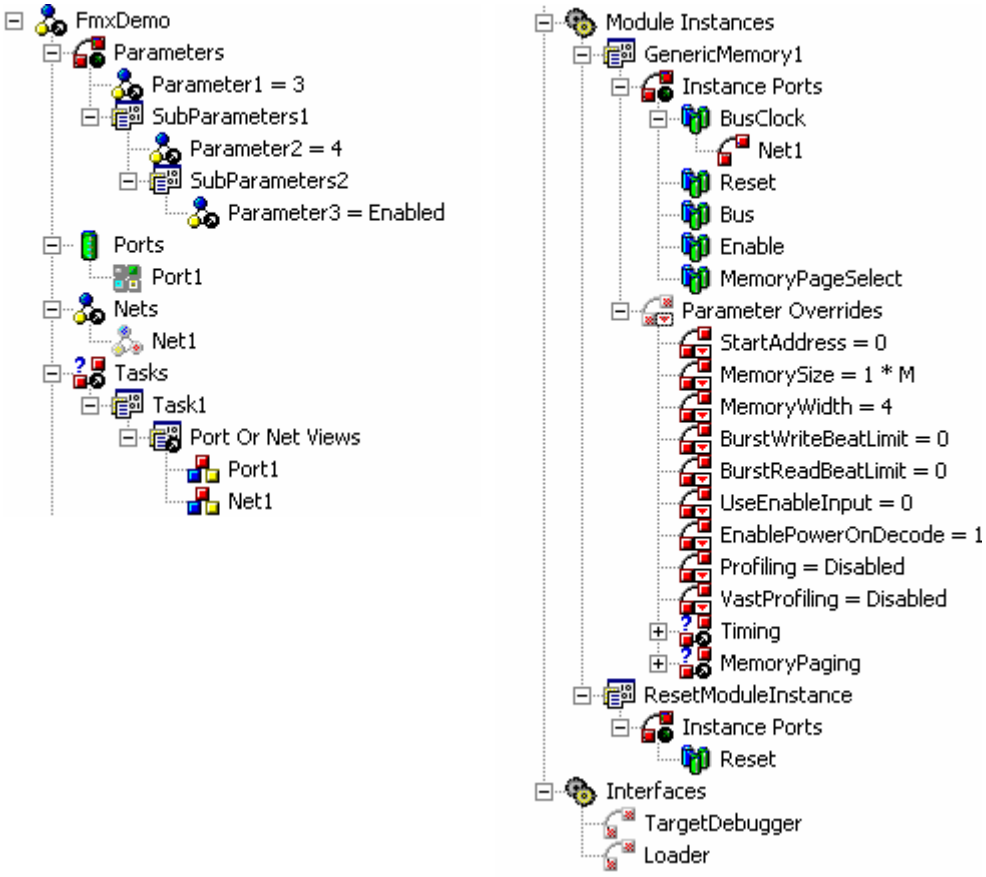
On choosing an Add or Edit action, the element dialog appears. Element dialogs have a standard format, with Properties listed, a control for editing each property, and the standard OK, Cancel, Apply and Help buttons at the foot of the dialog.



In the element descriptions below, dialog pictures are truncated to avoid repetition of standard elements.

FMX Hierarchy

Below is the XML Tree view of a .fmx file, showing all types of element.



VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly
authorized in a written agreement with VaST.

The table shows elements with their immediate child elements.

Element	Can Contain Child Elements
Fabric Module	Parameters, Ports, Nets, Tasks, Module Instances, Interfaces
Parameters	Parameter, Sub Parameters
Sub Parameters	Parameter, Sub Parameters
Parameter	*None*
Ports	Port
Port	*None*
Nets	Net
Net	*None*
Tasks	Task
Task	Port Or Net Views
Port Or Net Views	Port Or Net View (of specified Port or Net)
Port Or Net View	*None*
Module Instances	Module Instance
Module Instance	Instance Ports, Parameter Overrides
Instance Ports	Instance Port
Instance Port	Port Connection (to specified Port or Net)
Port Connection	*None*
Parameter Overrides	Parameter Override, Sub Parameter Overrides
Sub Parameter Overrides	Parameter Override, Sub Parameter Overrides
Interfaces	Interface
Interface	*None*

Hierarchical Parameter Type Naming

Parameter names should be chosen to distinguish parameters according to type. For example, if you have a number of memory modules, each with a parameter specifying the base address of the memory, you should give the parameter the same name in each case: e.g. `StartAddress`

Naming parameters according to type offers a number of advantages. When creating a Parameter Table View (see *Parameter Table Editing, page 34*) you can define a column using the common parameter type name. This allows you to view, compare and edit parameter values for all module instances sharing that parameter type.

Parameters with the same name can be distinguished by placing each in a separate subgroup.

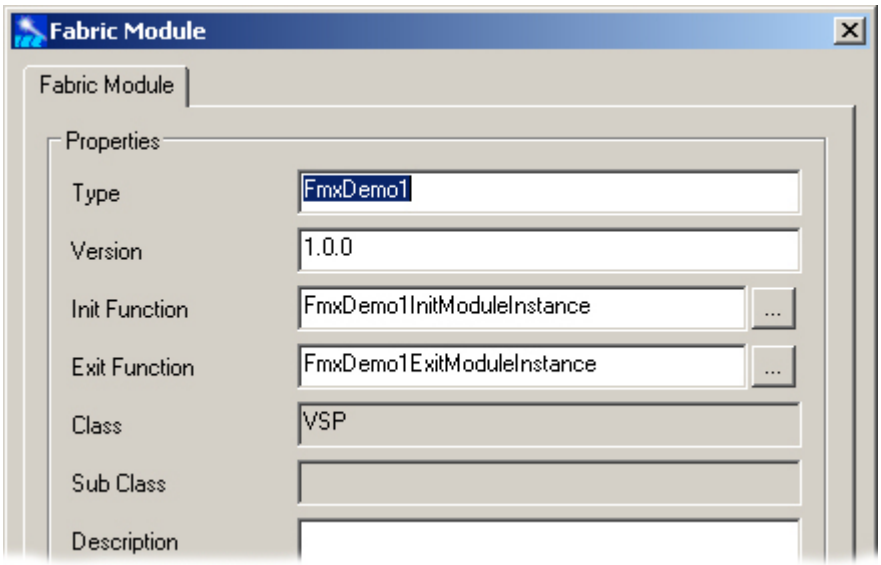
To name parameter types, for example when defining parameter table view columns, you can use a dot convention to specify a hierarchical name for parameters within Sub Parameters groups. For example, a parameter named Size belonging to the Sub Parameters group DataCache can be referred to as `DataCache.Size`. This distinguishes it from the parameter `InstCache.Size`, the Size parameter located in the InstCache Sub Parameters group.

When populating the parameter table view, for each module instance row, CoMET uses the first parameter it finds, within that module instance, matching the column parameter type.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Fabric Module

The Fabric Module element is the top level element of the Fabric Module Definition. Open it by double clicking the top level element in the XML Tree View. This element is labeled with its Type attribute in the XML Tree View.



Type: The type of module to be created. The module is listed under this type in the Module library, and in the Module Instance dialog, Type attribute list. Default: *ProjectName*.

Version: Specify the version of this module. This attribute is used in conjunction with the Module Instance Required Version attribute. See *Module Instance, page 84*.

Init Function: Specify the name of the function to be run by the simulator on initializing the module instance. This function is generated by the Add New Project wizard in `Source Files/`*`ProjectName`*`.c`. Click the button beside the function to view the source definition of the function.

Exit Function: Specify the name of the function to be run by the simulator on exit. Default name: *`ProjectName`*`ExitModuleInstance`. By default, a null definition of this function is generated by the Add New Project wizard in `Header Files/config.h`. Click the button beside the function to view the source definition of the function.

Class: This attribute is reserved for future use.

Sub Class: This attribute is reserved for future use.

Description: Provide a brief description of the Fabric Module. This description appears in Fabric Module Definition Specification reports.

Child Elements: Parameters, Ports, Nets, Tasks, Module Instances, Interfaces

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Parameters

A Fabric Module element may contain a single Parameters element. This element is a container for all Parameters and Sub Parameters elements. Sub Parameters elements allow you to set up nested groups of Parameters.

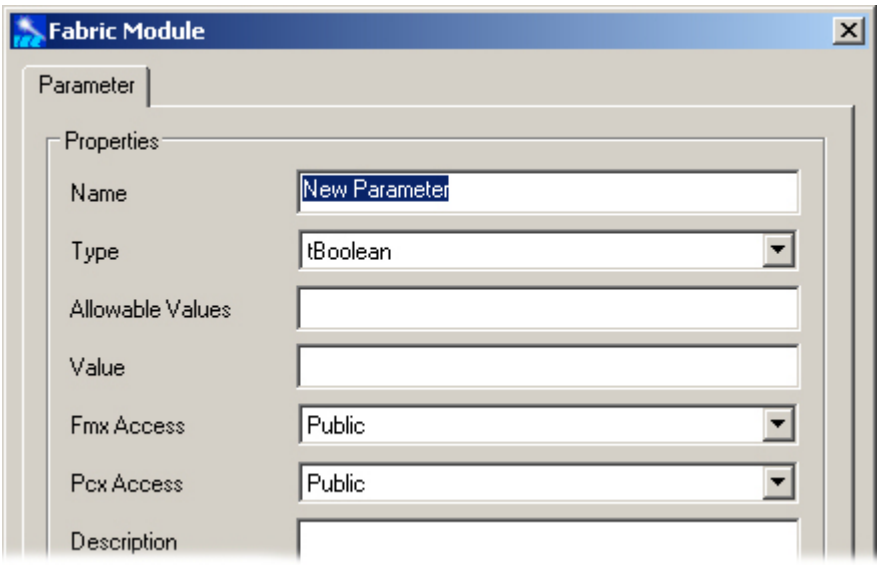
Child Elements: The Parameters element may contain one or more **Parameter** elements.

The Parameters element may contain one or more **Sub Parameters** elements.

Parameter

Parameters // Sub Parameter

Parameters allow you to create more general, configurable modules. When you create an instance of the module, corresponding Parameter Override elements are created in the Module Instance. Further Parameter Override elements are generated in the .pcx file for the containing VSP. The Parameter Override elements values can override the specified Parameter value. See *Parameter Override, page 89*.



Name: Type the desired name for the **Parameter**

Type: Select the type of the parameter. Allowed values: `tBoolean, tInt32, tInt64, tWord32, tWord64, tString, tAltList, tMultiList.` Types are defined in %vast5_home%\ampi\include\vast_std_types.h. To create a drop down menu, create a parameter of type tAltList.

Allowable Values: Specify allowable values for the parameter. Used only for **tAltList** and **tMultiList** types. Type a comma separated list of allowable values for the parameter. Validation by the Allowable Values list is implemented for parameters of type tAltList only.

Value: Type the default value for the parameter. This may be overridden by parameter overrides. Type the desired default value of the parameter. Values can be specified using macros and arithmetic e.g. `2*M, 24K-12`**.** If you leave **Value** blank, the corresponding Parameter Override generated in the .pcx file has the value `No Valid Value`. You must

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

provide a valid Parameter Override value for successful simulation. Parameter Values are not validated until simulation time.

Parameter Values can be modified in Parameter Views. See *Parameter Views, page 98*.

Fmx Access: Determines access to this parameter, via a Parameter Override, from a Fabric Module Definition (.fmx) file in which an instance of this module appears. Allowed values:

- `Public`: Visible and editable in fmx
- `Protected`: Visible, not editable in fmx
- `Private`: Not visible, not editable in fmx

See *Parameter Override, page 89*.

Pcx Access: Determines access to this parameter, via a Parameter Override, from a Prototype Configuration (.pcx) file in which an instance of this module appears. Allowed values:

- `Public`: Visible and editable in pcx
- `Protected`: Visible, not editable in pcx
- `Private`: Not visible, not editable in pcx

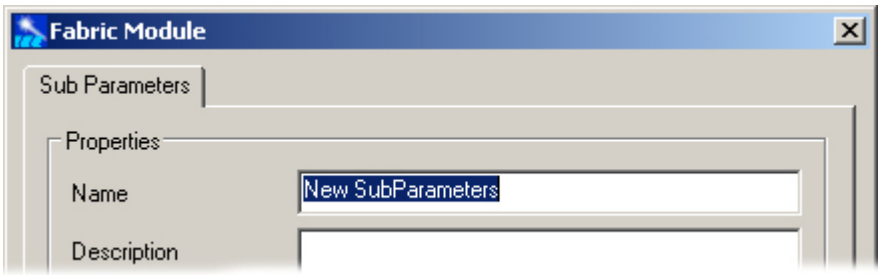
See *Parameter Override, page 89*.

Description: Provide a brief description of this Parameter.

Sub Parameters

Parameters // Sub Parameters

Sub Parameters provide a way hierarchically to distinguish parameters. See *Hierarchical Parameter Type Naming, page 76*.



Name: Type the desired name for the **Sub Parameters** element. This name acts as a category heading for parameter child elements.

Description: Provide a brief description of this **Sub Parameters** element.

Child Elements: A **Sub Parameters** element may contain one or more **Parameter** elements.

A **Sub Parameters** element may contain one or more **Sub Parameters** elements.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Ports

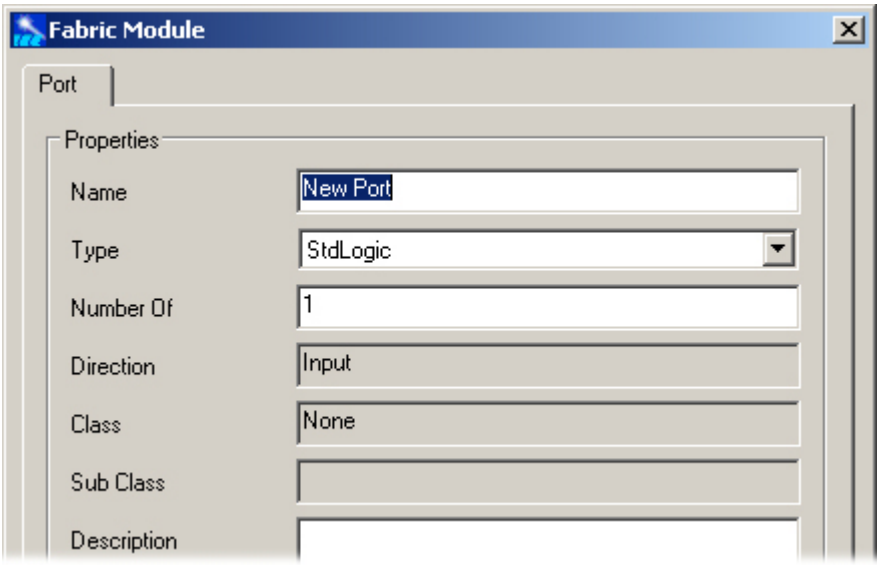
A Fabric Module element may contain a single Ports element. This element is a container for all **Port** elements.

Child Elements: The Ports element may contain one or more **Port** elements.

Port

Ports / Port

A Port is a connection point for connecting an instance of the current Fabric Module to a Port element of the parent platform, to Port elements of child Module Instances or to Net elements within the same platform. A Port has no child elements.



Fabric Module dialog, **Port** tab, **Properties** panel:

Name: Type a name for the port. See *Choosing Names, page 67*. You can create a Port array. See *Arrays, page 67*.

Type: Select a net type. The Port and any Net or Port to which it connects must be the same net type. Allowable Values: `StdBus`, `StdClock`, `StdLogic`, `CanBus`, `StdVector32`

Number Of: You can create a Port array by specifying a dimension in this attribute. Each member of the array shares properties but may be independently connected. See *Arrays, page 67*.
Allowable values: Positive integer Default: 1: A single Port, not an array.

Direction: To be used for assisted connection in future. The value is currently ignored.

Class: Reserved for future use.

Sub Class: Reserved for future use.

Description: You may add a description for documentation purposes.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Nets

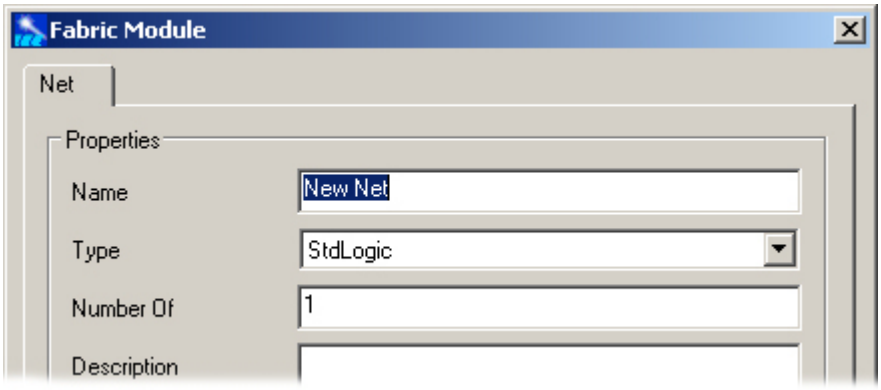
A Fabric Module element may contain a single Nets element. This element is a container for all **Net** elements.

Child Elements: The Nets element may contain one or more **Net** elements.

Net

Nets / Net

A Net allows a connection between Ports of two Module Instances within a platform.



Name: Type a name for the Net. See *Choosing Names, page 67.* You can create a Net array. See *Arrays, page 67*.

Type: Select a net type. The Net and any Port to which it connects must be the same net type. Allowable Values: `StdBus, StdClock, StdLogic, CanBus, StdVector32`

The `Connect` Net Type is reserved for VaST internal use.

Number Of: You can create a Net array by specifying a dimension in this attribute. Each member of the array shares properties but may be independently connected. See *Arrays, page 67*.

Allowable values: Positive integer Default: 1: A single Port, not an array.

Description: You may add a description for documentation purposes.

Child Elements: None

Tasks

A Fabric Module element may contain a single Tasks element. This element is a container for all **Task** elements.

Child Elements: The Tasks element may contain one or more **Task** elements.

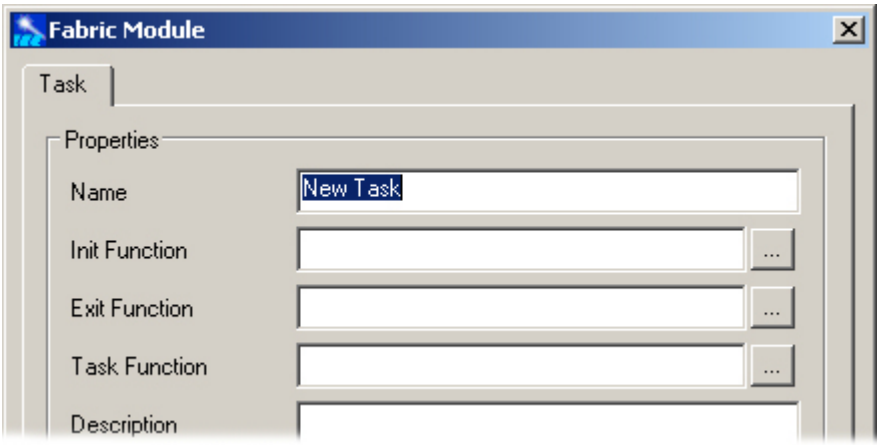
Task

Tasks / Task

You can model behavior with tasks or callback functions. Callbacks run in response to events. Tasks are initialized and run after module initialization. Details of callbacks and tasks are discussed in *The CoMET Modeling Guide (CIF)*.

When you create a top level VSP module typically you do not have to add tasks. When you create a module of type Peripheral Module some default tasks are created in the .fmx file and the skeleton module code (*projectName*.c file). If you want to add an additional task to a peripheral module, you need both to declare it in the module .fmx and to create the task code. Declaring the task in the .fmx file notifies the simulator of the existence of the task.

You do not need to specify a name for the Init Function, Exit Function or Task Function attributes. CoMET assigns a default name for these default functions, as indicated below. The name must match the name in the source code for the module. If the named task functions do not appear in the module source code, CoMET displays errors on Build.



Name: Type a name for the Task. For more on Tasks see the *CoMET Modeling Guide (CIF)*

Init Function: Type a name or leave blank to allow CoMET to assign default `TaskNameInitTaskInstance`. Click the button beside the function to view the source definition of the function.

Exit Function: Type a name or leave blank to allow CoMET to assign default `TaskNameExitTaskInstance`. Click the button beside the function to view the source definition of the function.

Task Function: Type a name or leave blank to allow CoMET to assign default `TaskNameTaskFunction`. Click the button beside the function view the source definition of the function.

Description: You may add a description for documentation purposes.

Child Elements: A Task can contain a single **Port Or Net Views** element.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Port Or Net Views

Task/Port Or Net Views

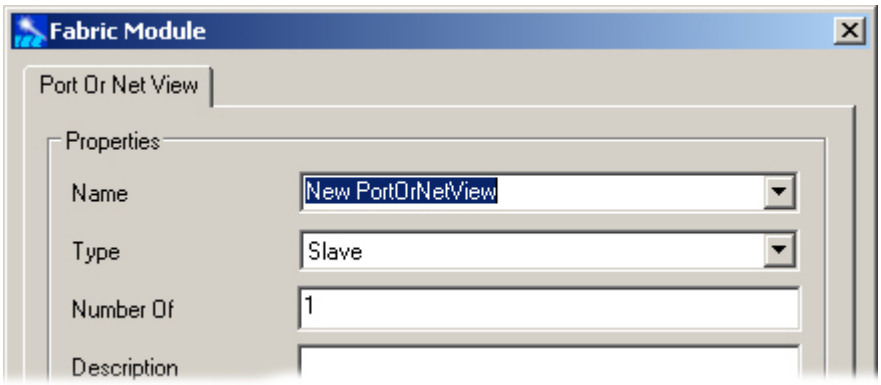
A Task may contain a single **Port Or Net Views** element. This element is a container for all **Port Or Net View** elements belonging to the Task.

Child Elements: The Port Or Net Views element may contain one or more **Port Or Net View** elements.

Port Or Net View

Task/Port Or Net Views/Port Or Net View

A Port Or Net View specifies the task access to the Port or Net.



Name: Select a Port or Net from the list. Before creating a Port Or Net View you must first create the corresponding Port or Net. You can refer to members of a Port or Net array using array notation. See *Arrays, page 67*.

Type: Select a Port Or Net View type. Allowable Values:

- `Slave`: The Task provides a response to a stimulus on the Net or Port. A slave connection typically limits a task to reading values of the Net or Port.
- `Master`: The Task provides a stimulus to the Net or Port. The task can drive values onto the Net or Port.
- `MasterSlave`: The Task both provides a stimulus to and responds to a stimulus on the Net or Port
- `NetController`: This is currently used only by VaST built Net Control Protocol Engines.
- `NetSlave`: Equivalent to Slave. Provided for compatibility. Use Slave instead.
- `NetMaster`: Equivalent to Master. Provided for compatibility. Use Master instead.

Number Of: This attribute may refer to members of a Port or Net array. Use array notation, omitting brackets. e.g. `0:15` or `0,1,2` Array notation in the **Name** attribute takes precedence over this field. See *Arrays, page 67*. Default: 1, a single element, not an array.

Description: You may add a description for documentation purposes.

Module Instances

A Fabric Module of the Virtual System Prototype or Virtual Platform class may contain a single Module Instances element. This element is a container for all **Module Instance** elements.

Child Elements: The Module Instances element may contain one or more **Module Instance** elements.

Parameter Override elements also appear in the Virtual System Prototype (VSP) Configuration (.pcx) file defined for the VSP. The Prototype Configuration file contains Module Instances and their Parameter Override elements, for every Module Instance within the VSP hierarchy.

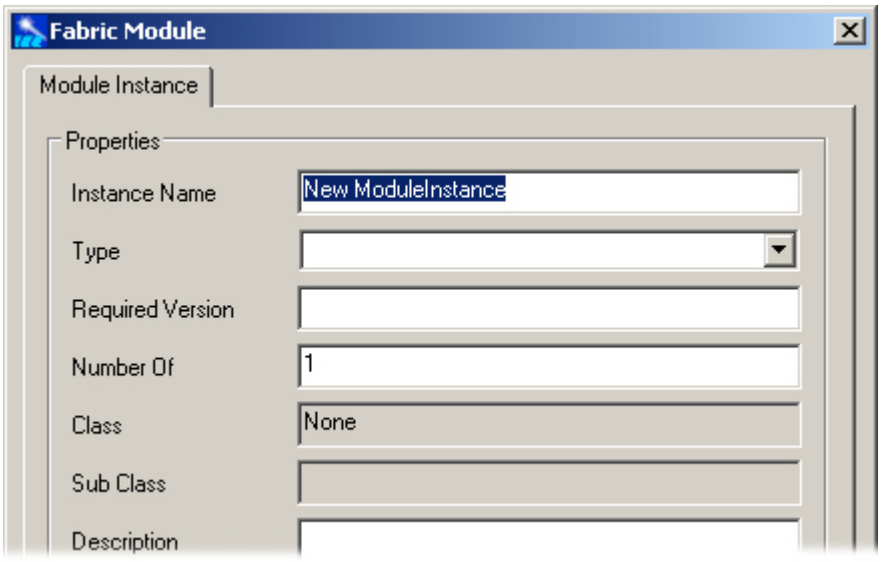
Module Instance

Module Instances / Module Instance

You can instantiate a module any number of times either by creating multiple instances, each with separate Module Instance Names, or by creating a Module Instance specified as an array.

There are several ways to add Module Instances. You can use the Module Instances context menu option, or use the Modules Window to add modules using drag and drop or cut and paste.

Module Instance elements also appear in the Virtual System Prototype (VSP) Configuration (.pcx) file defined for the VSP. The Prototype Configuration file contains Module Instances and their Parameter Override elements, for every Module Instance within the VSP hierarchy. See *Parameter Override, page 89*.



VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Instance Name: Type the desired Instance Name. Each Instance Name must be unique. You can create a Module Instance array. See *Arrays, page 67*.

Module Type: Select one of the listed module types. Listed modules are either projects within the Workspace or compiled modules in the Modules library. See *Modules Configuration Settings, page 38*.

Required Version: This attribute is used in module versioning, in conjunction with the Fabric Module Version attribute. If a Required Version is specified, on simulation CoMET attempts to locate and load a module of the specified version. If a module with the specified Required Version cannot be found, CoMET displays an error. If no Required Version is specified, CoMET locates and loads the latest version of the module.

To locate a module version, CoMET first searches the current workspace, then the `%vast5_home%\Modules` directory, and then any additional user defined module library directories.

Number Of: You can create a Module Instance array by specifying a dimension in this attribute. Each member of the array shares properties but may be independently connected. See *Arrays, page 67*.

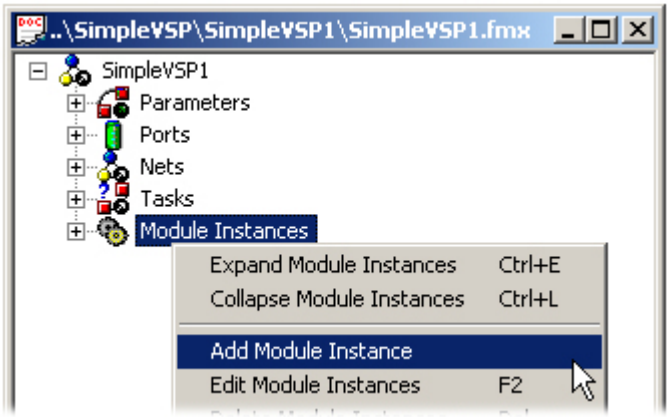
Description: You may add a description for documentation purposes.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

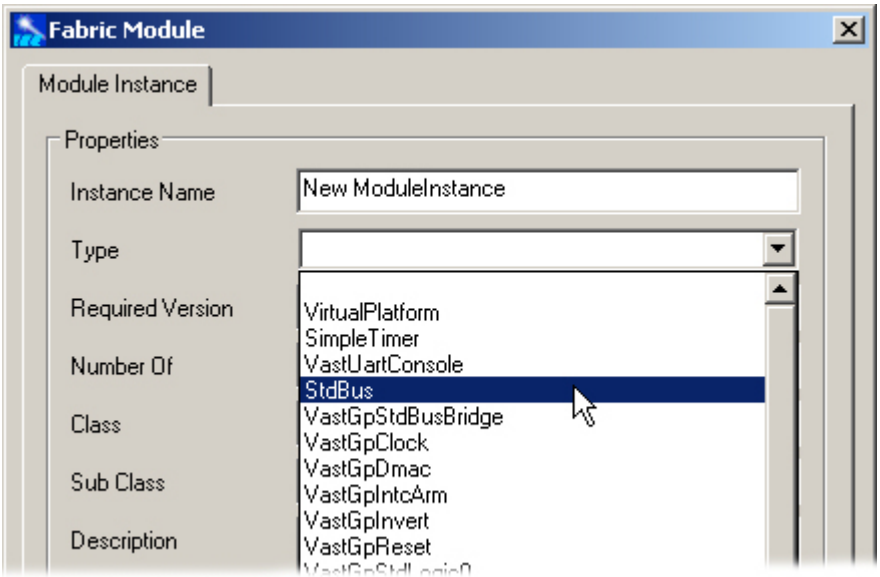
Adding a Module Instance using the Context Menu

To add module instances to a module using the context menu:

- In the fmx XML Tree View, right click on **Module Instances**.
- Select Add Module Instance.



In the Fabric Module/Module Instance dialog, select the Type from the Type combo box list. This list shows the available modules in the current Workspace, in the VaST module library and in any user module libraries.

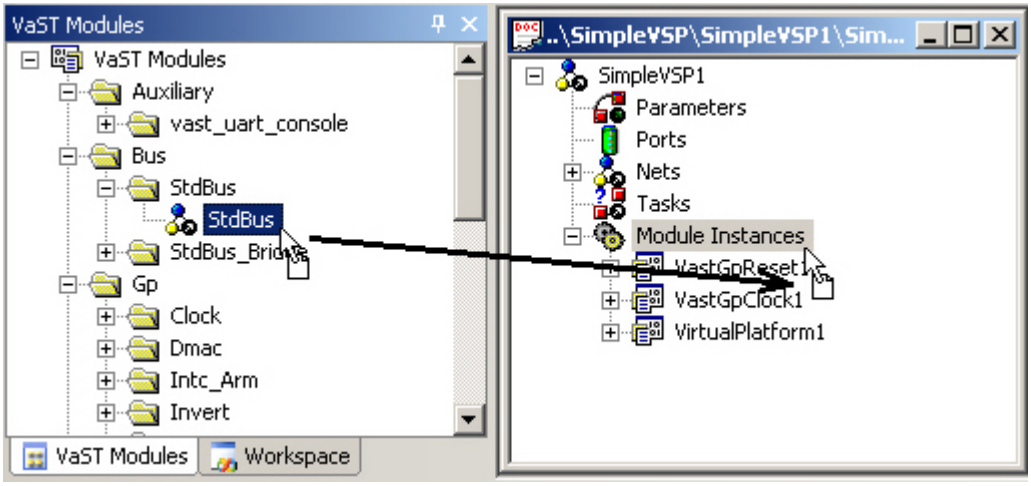


VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

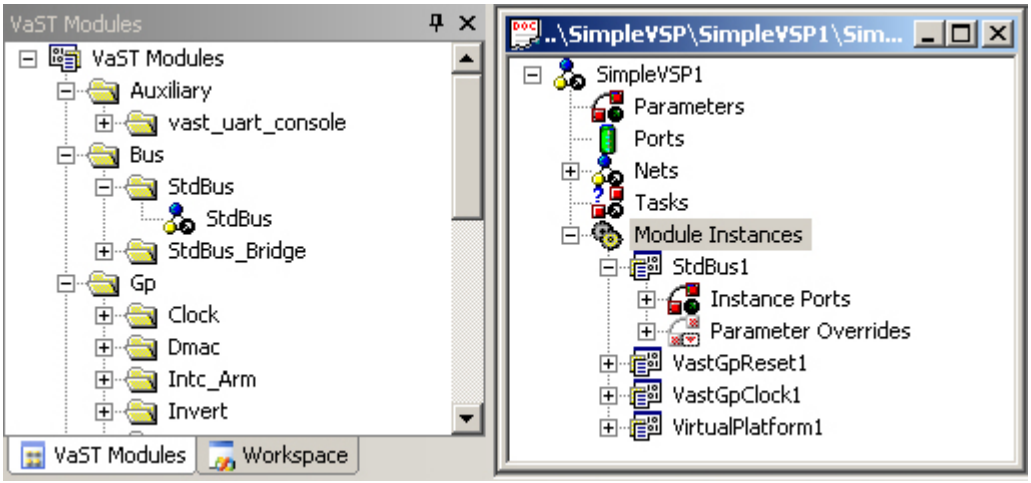
Adding a Module Instance using Drag and Drop from the Modules window

To add module instantiations to a module using the Modules Window using drag and drop:

- Open the required .fmx file in XML Tree View.
- In the Modules Window, navigate to the icon for the module of which you want to add an instance.
- Click on the module icon, drag it to the **Module Instances** node in the .fmx file, and drop.



The new module instance appears in the .fmx file view, with a default name.



VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Adding a Module Instance using Copy and Paste

To add module instantiations to a module using copy and paste from the Modules Window:

- Highlight the module icon in the Modules window
- Copy (Ctrl-C, Edit/Copy)
- Highlight module instances in the .fmx XML tree view
- Paste (Ctrl-V, Edit/Paste)
- You can also copy a module from the Modules window and paste into the appropriate .fmx file table view. For example, copy VastGpSTdBusBridge into the .fmx file Bus Connections view.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly
authorized in a written agreement with VaST.

Parameter Overrides

Module Instances / Module Instance / Parameter Overrides

A Module Instance element may contain a single Parameter Overrides element. This element is a container for all Parameter Overrides and Sub Parameter Overrides elements within the Module Instance.

Child Elements: The Parameter Overrides element may contain one or more **Parameter Override** elements.

The Parameters element may contain one or more **Sub Parameter Overrides** elements.

Parameter Override

Module Instances / Module Instance / Parameter Overrides / Parameter Override

When you create a parameter, it is defined and its default value is set in the Fabric Module Definition (.fmx) file. When you add a Module Instance of that Module to another Fabric Module, CoMET examines each Parameter element defined in the Module of which you are adding an instance. For a Parameter element whose Fmx Access attribute has a value of Public or Protected, CoMET creates a corresponding Parameter Override element. It maintains the Parameter hierarchy by creating Sub Parameter Override elements corresponding to each Sub Parameters element in the module.

You can override a Module Instance Parameter value by altering the values in the corresponding Parameter Override elements.

You cannot delete Parameter Override elements, or add them. The Edit feature in the Parameter Override context menu allows you directly to change the Parameter Override value without opening the element dialog.

Parameter Override elements also appear in the Virtual System Prototype (VSP) Configuration (.pcx) file defined for the VSP. The Prototype Configuration file contains Module Instances and their Parameter Override elements, for every Module Instance within the VSP hierarchy. A Prototype Configuration Parameter Override value takes precedence over the matching platform fmx file Module Instance Parameter Override value, which in turn takes precedence over the matching module fmx file Parameter Value defined when creating the module.

Alterations to a module must result in changes to module instances of that type. See *Fmx Update, page 72*.

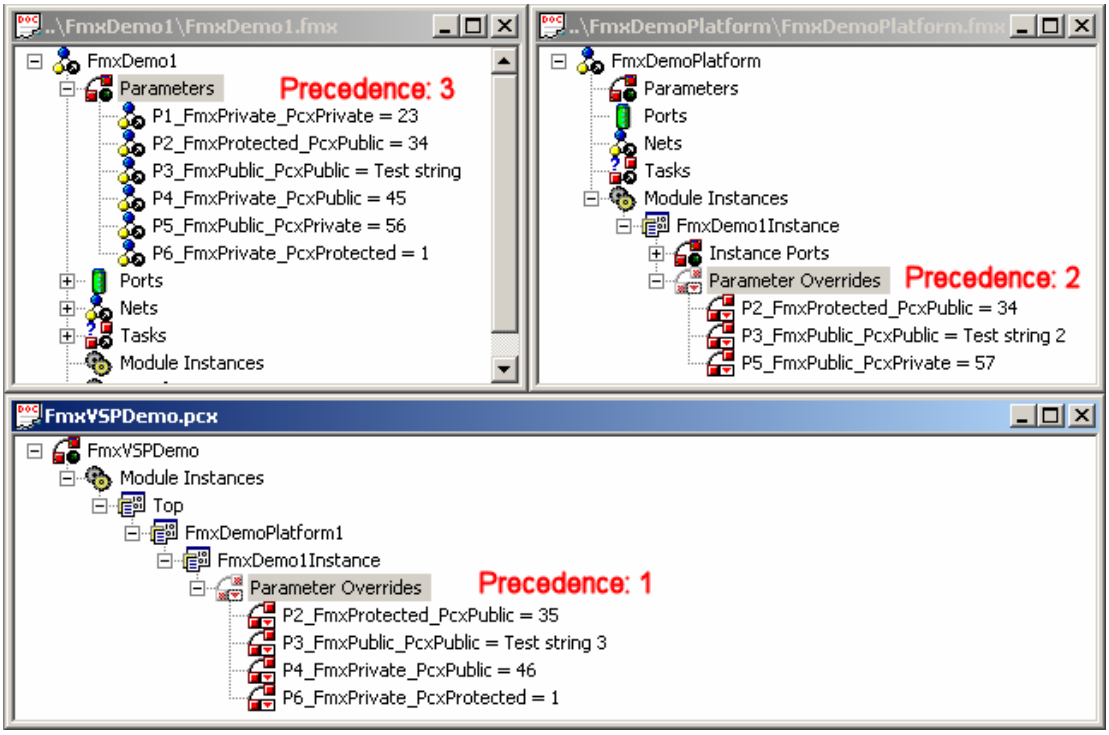
VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Parameter Fmx Access, Pcx Access and Parameter Overrides

The picture below shows examples of how varying the Fmx Access and Pcx Access in Module Parameters affects the Parameter Overrides of a corresponding Module Instance.

FmxDemoPlatform contains an instance of the FmxDemo1 module. FmxVSPDemo contains an instance of FmxDemoPlatform.

This results in Parameter Overrides in FmxDemo1Instance in FmxDemoPlatform.fmx, and in FmxDemo1Instance in FmxVSPDemo.pcx, corresponding to the Parameters in FmxDemo1.fmx.



FmxDemoPlatform.fmx displays only parameters with Fmx Access set to Public or Protected.

FmxVSPDemo.pcx file displays only parameters with Pcx Access set to Public or Protected.

The final values for the FmxDemo1Instance Parameters are:

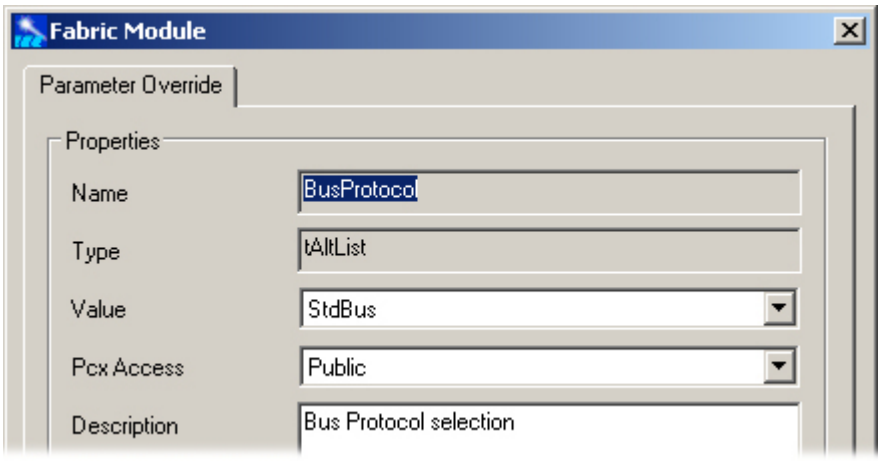
- P1_FmxPrivate_PcxPrivate = 23
- P2_FmxProtected_PcxPublic = 35
- P3_FmxPublic_PcxPublic = Test string 3
- P4_FmxPrivate_PcxPublic = 46
- P5_FmxPublic_PcxPrivate = 57
- P6_FmxPrivate_PcxProtected = 1

For more on the Prototype Configuration (.pcx) file, see *Modifying Parameter Values in the Prototype Configuration File, page 121*.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

The Pcx Access Field is Always Editable in a Parameter Override

A module designer may set Fmx Access to Protected for a module Parameter. A platform designer may create a module instance of the module. To permit the platform designer to control access to the corresponding module instance Parameter Override, the Pcx Access field of the corresponding Parameter Override is editable, even though the Parameter has Fmx Access set to Protected.



You can view the attributes of a Module Instance Parameter Override, but the only attribute you can change is Value. If the original Parameter has Protected access, the Value cannot be edited.

Name: This name matches a module Parameter name.

Type: This name is derived from the Type attribute of the corresponding module Parameter.

Value: This contains the value that overrides the corresponding Parameter value. Default: Corresponding Parameter value.

Pcx Access: Determines access to this parameter, via a Parameter Override, from a Prototype Configuration (.pcx) file in which an instance of this module appears. Allowed values:

- `Public`: Visible and editable in pcx
- `Protected`: Visible, not editable in pcx
- `Private`: Not visible, not editable in pcx

See *Parameter Fmx Access, Pcx Access and Parameter Overrides* above.

Parameter Override Values can be modified in Parameter Views. See *Parameter Views, page 98*.

Description: This description is derived from the description attribute of the corresponding module Parameter.

Changing the Override Value

To alter the Override Value of a Parameter Override element,

Either:

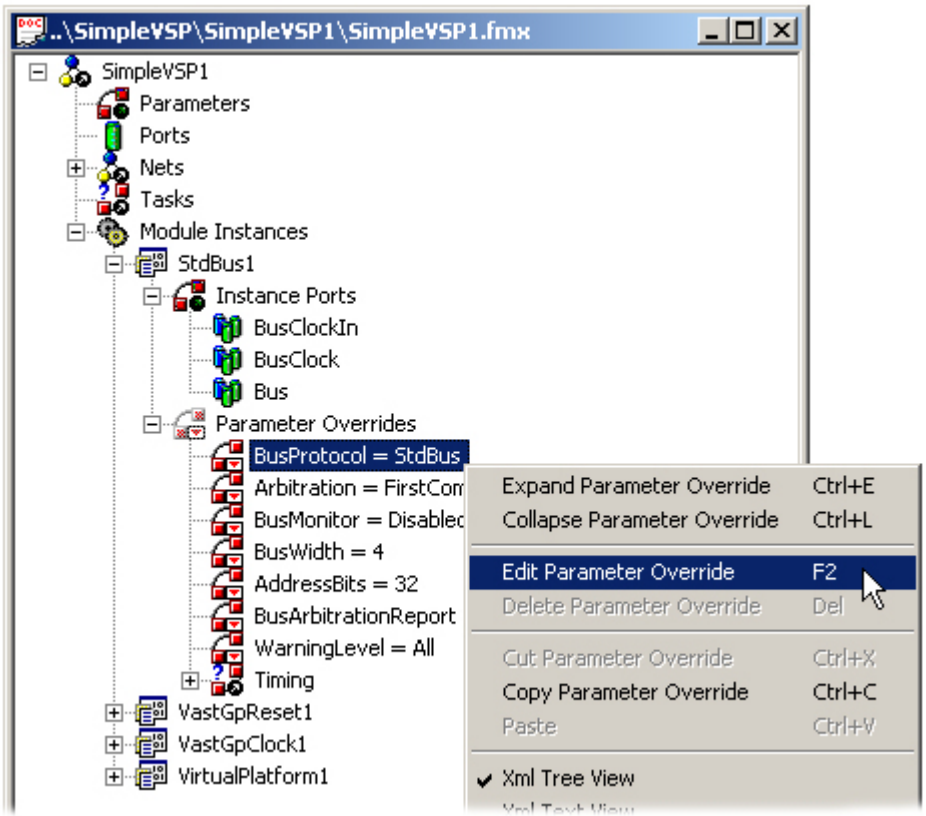
- Double click on the **Parameter Override** element (or otherwise open the Parameter Override dialog)
- Alter the Override Value attribute

Alternatively:

- In the .fmx or .pcx XML Tree View, expand the Module Instance and the Parameter Overrides nodes.
- Right click on the parameter you wish to edit.
- Select **Edit Parameter Override** from the Context menu.
- Edit the value of the parameter override and press Enter to complete.

Alternatively

- Select the parameter in the XML Tree View and press F2.
- Edit the value of the parameter override and press Enter to complete.



VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

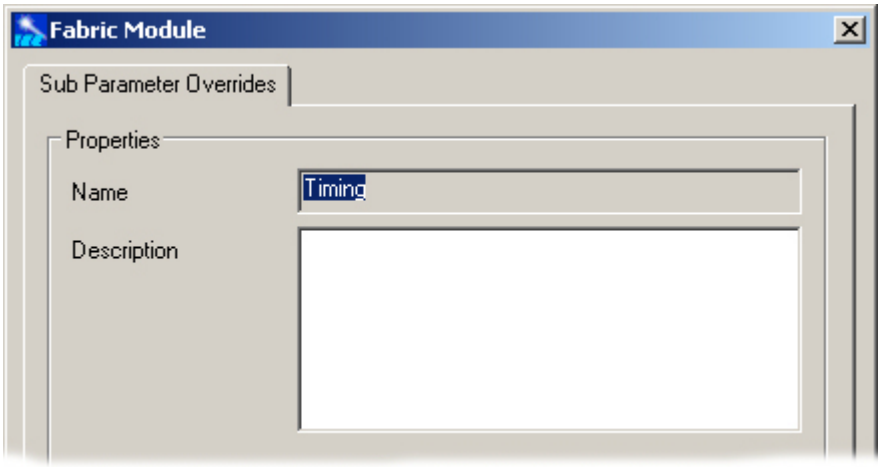
Sub Parameter Overrides

Module Instances / Module Instance // Sub Parameter Overrides

A Sub Parameter Overrides element corresponds to a Sub Parameters element in the module from which the module instance derives. It maintains the hierarchy of the Sub Parameters and Parameter elements in the module, allowing correct placement of Parameter Override elements.

When you add a Module Instance to a Fabric Module, CoMET automatically creates Sub Parameter Override elements corresponding to the Parameter Override elements in the module.

Sub Parameter Overrides also appear in the VSP Prototype Configuration (.pcx) file.



Name: This matches the name of the Sub Parameters element in the module. Its path of parent Sub Parameters elements must also match the path of Sub Parameters elements in the module. A Sub Parameters element has no purpose if its name does not match a module Sub Parameters element.

Child Elements: The Sub Parameter Overrides element may contain one or more **Parameter Override** elements. The Sub Parameter Overrides element may contain one or more **Sub Parameter Overrides** elements.

Description: You may add a description for documentation purposes.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Instance Ports

Module Instances / Instance Ports

A Module Instance element may contain a single Instance Ports element. This element is a container for all Instance Port elements within the Module Instance. When you add a Module Instance to a Fabric Module, CoMET automatically creates an Instance Ports element and Instance Port elements within it, corresponding to all the Port elements in the module.

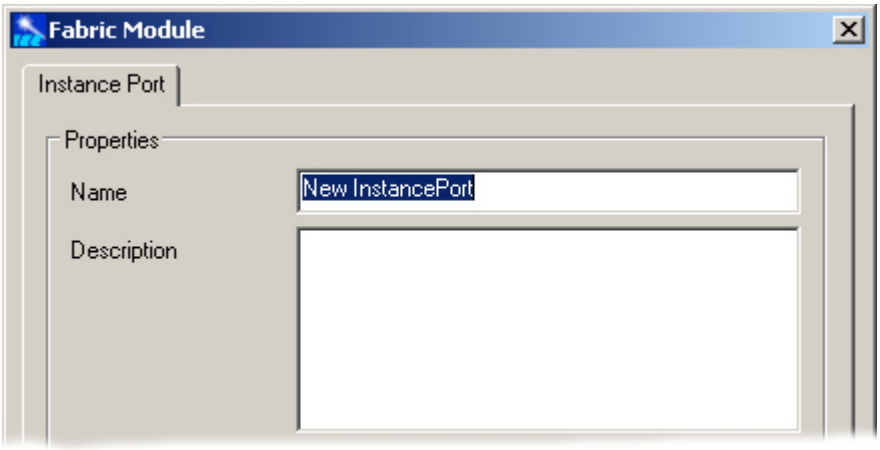
Child Elements: The Instance Ports element may contain one more **Instance Port** elements.

Instance Port

Module Instances / Instance Ports / Instance Ports

An Instance Port in a Module Instance corresponds to a Port in the module. The Instance Port can contain a Port Connections element, which in turn contains Port Connection elements, allowing multiple connections between the Instance Port and Port elements or Net elements within the parent Platform.

Alterations to a module must result in changes to module instances of that type. See *Fmx Update, page 72*.



Name: This must match a Port in the module. You can refer to members of an Instance Port array. See *Arrays, page 67*.

Description: You may add a description for documentation purposes. Descriptions appear in Prototype Reports. See *Viewing Module Definition and Prototype Configuration Reports, page 100*.

Child Elements: An Instance Port may contain one or more Port Connection elements.

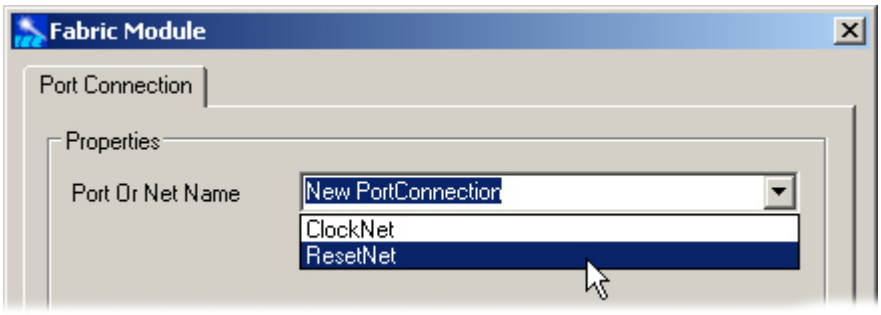
VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Port Connection

Module Instances / Instance Ports / Instance Ports / Port Connection

A Port Connection element specifies a connection between a **Module Instance / Instance Port** and a **Ports / Port** or **Nets / Net** element of the containing platform.

Alterations to a module must result in changes to module instances of that type. See *Fmx Update, page 72*.



Port Or Net Name: Select from the list the Port or Net to which you wish to connect the Instance Port. CoMET lists all Port and Net elements belonging to the Fabric Module. You can refer to members of a Port or Net array. See *Arrays, page 67*.

Port Connection Port Or Net Name values can be modified in a Connection View. See *Connection Views, page 99*.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Interfaces

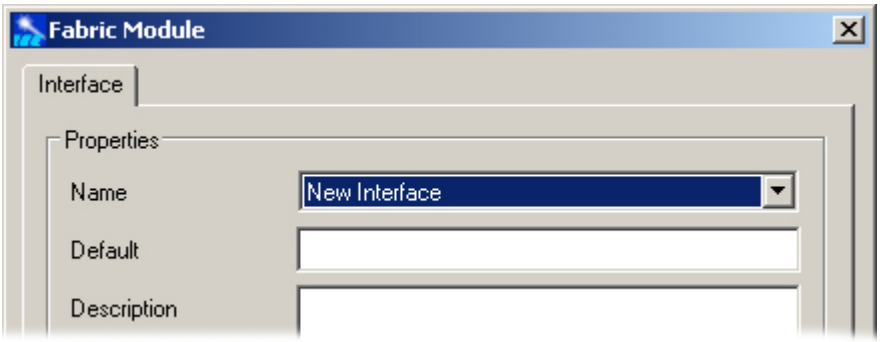
A Fabric Module may contain a single Interfaces element. This element is a container for all **Interface** elements.

Child Elements: The Module Instances element may contain one or more **Interface** elements.

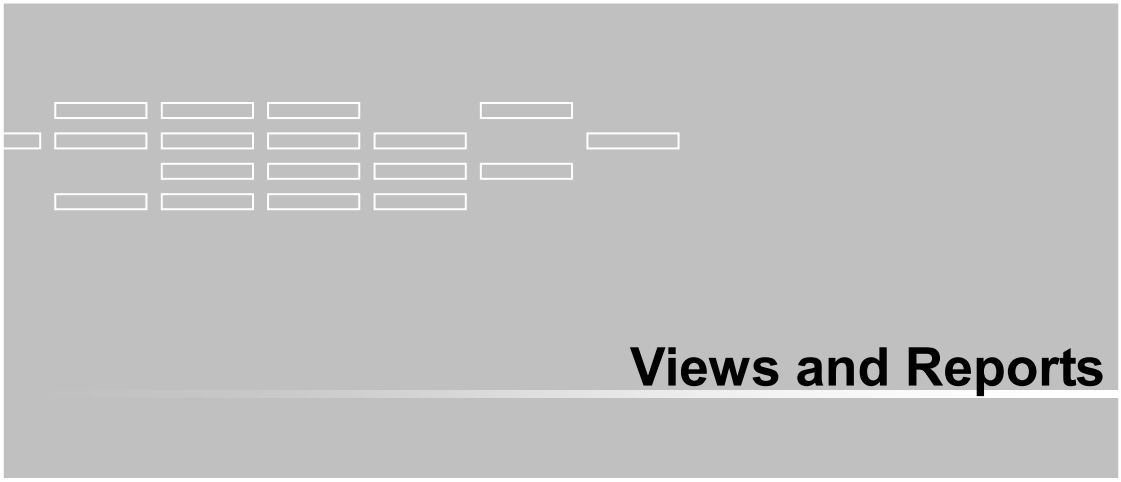
Interface

Interfaces / Interface

The Interfaces element is used only when developing either modules that have direct access to the target debugger, or modules that require a target image loader. Typically these modules are Virtual Processor Models (VPMs) or memory modules. More information on these features will be provided in future.



VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly
authorized in a written agreement with VaST.



Views and Reports

You can view information on your Fabric Module in a number of different ways. CoMET provides a number of different views and reports.

Using Table Views

Table views allow you to view the .fmx file as a filtered list in various formats. The table views offer ways to view and edit the .fmx file, as alternatives to using the XML Tree View.

To choose a table view, right click on a line in the .fmx document window and select the view from the context menu. The table view shows all specified parameters or connections for the project. If you press the Ctrl key while right clicking on a module instance node, the table is restricted to the specified parameters or connections within the selected module instance

You can configure table views to show the information you require. See XML Tables Configuration Settings*, page 33*. You can specify the default view you prefer on opening an .fmx or .pcx file. See *General Configuration Settings, page 26*. CoMET remembers the current view of each open document when you close a workspace, so when you open the workspace again the document appears in the last view chosen.

Table views are of two main types:

- Parameter views
- Connection views

These types are discussed in detail below.

Changing Column Order

You can drag and drop a column label to change the order of columns. This is useful, for example, if you wish to position the InstanceName column next to a column on the far right. Changes to column order are not preserved when CoMET is closed.

Parameter Views

A parameter view shows a list of parameters, filtered on parameter type. For each module instance there is a row in the table, identified by the InstanceName in the left column. The second column shows ModuleType. The remaining table columns are the parameter types specified in the XML table configuration. See *XML Tables Configuration Settings, page 33*.

The **Memory Parameter View** shows a list of all parameters belonging to the module or its submodules, filtered on the parameter types:

- Read
- Write
- StartAddress
- MemorySize
- MemoryWidth
- Tunneling
- NoOfMemoryPages
- PageSize

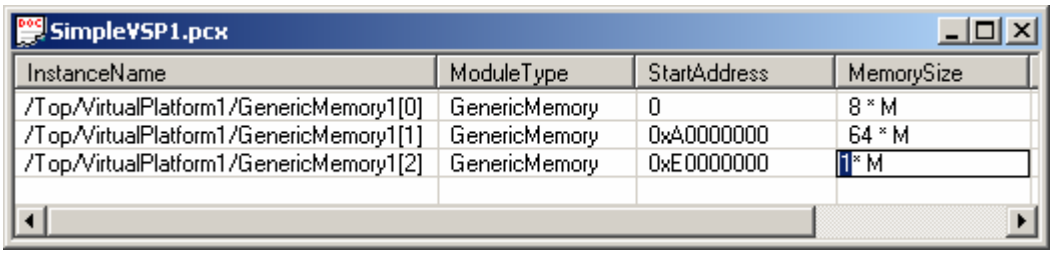
The **Bus Parameter View** shows a list of parameter belonging to the modules or its submodules, filtered on the parameter types:

- BusProtocol
- Arbitration
- BusMonitor
- BusWidth
- AddressBits
- RequestLatency

The **Vpm Parameter View** shows a list of parameter belonging to the modules or its submodules, filtered on the parameter types:

- Endian
- Metrix
- WorkBufferSize
- Debugger
- MinimumOil
- DynamicOil
- DataCache.Size
- DataCache.Ways
- InstCache.Size
- InstCache.Ways

VaST modules follow a standard parameter naming scheme (see *Hierarchical Parameter Type Naming, page 76)* with parameters named according to the types above, so the Memory Parameter View may list a number of modules and show their corresponding parameter values.



You can change the value of a parameter in a Parameter View by clicking on the cell displaying the parameter value and editing it. You can also use parameter views to edit .pcx files. The picture shows a .pcx file being edited in Memory Parameter View.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Connection Views

A connection view shows module instance port or net connections, filtered on the type of port or net. For each Module Instance there is a row in the table, identified by the InstanceName in the first column on the left. The second column shows the ModuleType. The remaining table columns represent the module ports and nets. Ports are followed by a `.port` type suffix. Nets are followed by a `.net` type suffix. An entry in a cell shows a connection between a port belonging to the row module instance, and the column net or port.

NOTE: a connection view can show only one connection where:
- multiple ports from the same device connect to the same net or port
- a single instance port is connected to multiple nets or ports

The possible net and port types are:

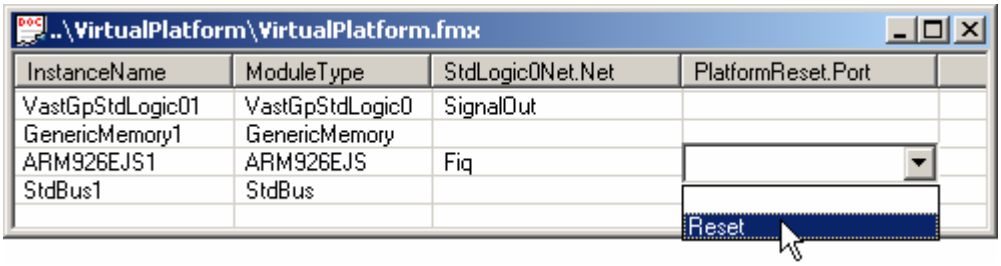
- StdBus
- StdClock
- StdLogic
- StVector32

The default connection views show both nets and ports, filtered on each of the types:

- Bus Connection View: StdBus
- Clock Connection View: StdClock
- Logic Connection View: StdLogic

You can configure connection views according to your requirements. See XML Tables Configuration Settings, *page33*.

To edit connections in connection view, click in the cell at the intersection of the required module instance row and the required net or port column. CoMET displays a list of available ports. Select the port to connect the selected module instance port to the selected net or port.



The picture shows an .fmx file being edited in Logic Connection View. The Reset Port of the ARM926EJS1 module instance is being connected to the PlatformReset module port. Note that the columns are identified as Net or Port by a suffix.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Viewing Module Definition and Prototype Configuration Reports

CoMET provides a number of different views of module structure and prototype configuration. The XML Tree view and XML Table views allow you to browse and edit Fabric Module Definition (.fmx) files and Prototype Configuration (.pcx) files within the Document window.

You can also view a comprehensive report on the contents of an .fmx or a .pcx file, using the Prototype Report feature. To view Prototype Reports, you need an XML viewer capable of performing XSL transformations. Recent versions of Microsoft Internet Explorer (http://www.microsoft.com/windows/ie) and Mozilla (http://www.mozilla.org) have this capability, and may be freely downloaded and used.

NOTE:	Prototype reports open in the application specified for performing the Open action on .xml files. This must be a browser capable of viewing HTML and of performing XSL transformation for prototype reports to appear in the correct format. To determine the application registered to open .xml files, open Windows Explorer, choose Tools/Folder Options/File Types and browse to the XML entry.

To view a Prototype Report:

- Open a .pcx or .fmx file in the Document windows and choose **Tools/Prototype Report**,

or

- click the Prototype Report button on the tool bar.

Viewing a Fabric Module Definition Report - Fmx Report

Open the Fabric Module Definition (.fmx) file in the Document Window.

To open an .fmx file in the Document Window:

- Double click the .fmx file icon or name in the Workspace window

or

- To open the .fmx of a module in the module library, double click on the module icon

or

- Open a Fabric Module .dll by choosing **File/Open File** and browsing for the .dll file.

To view the prototype report for the .fmx file

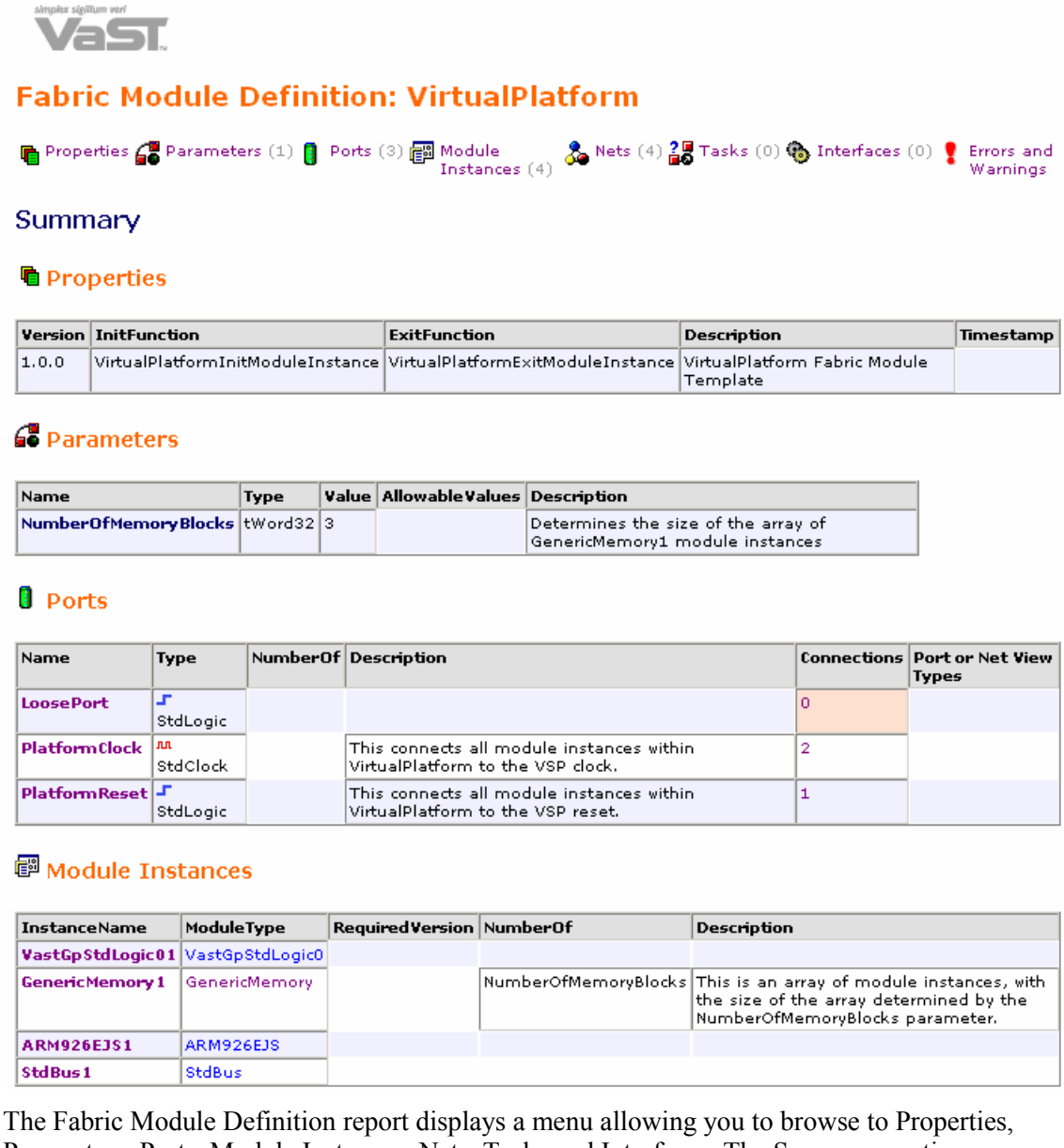
- Select **Tools/Prototype Report**

or

- click the Prototype Report button .

CoMET opens the report in the default XML browser.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.



Fabric Module Definition: VirtualPlatform

Properties | Parameters (1) | Ports (3) | Module Instances (4) | Nets (4) | Tasks (0) | Interfaces (0) | Errors and Warnings

Summary

Properties

Version	InitFunction	ExitFunction	Description	Timestamp
1.0.0	VirtualPlatformInitModuleInstance	VirtualPlatformExitModuleInstance	VirtualPlatform Fabric Module Template	

Parameters

Name	Type	Value	AllowableValues	Description
NumberOfMemoryBlocks	tWord32	3		Determines the size of the array of GenericMemory1 module instances

Ports

Name	Type	NumberOf	Description	Connections	Port or Net View Types
LoosePort	StdLogic			0	
PlatformClock	StdClock		This connects all module instances within VirtualPlatform to the VSP clock.	2	
PlatformReset	StdLogic		This connects all module instances within VirtualPlatform to the VSP reset.	1	

Module Instances

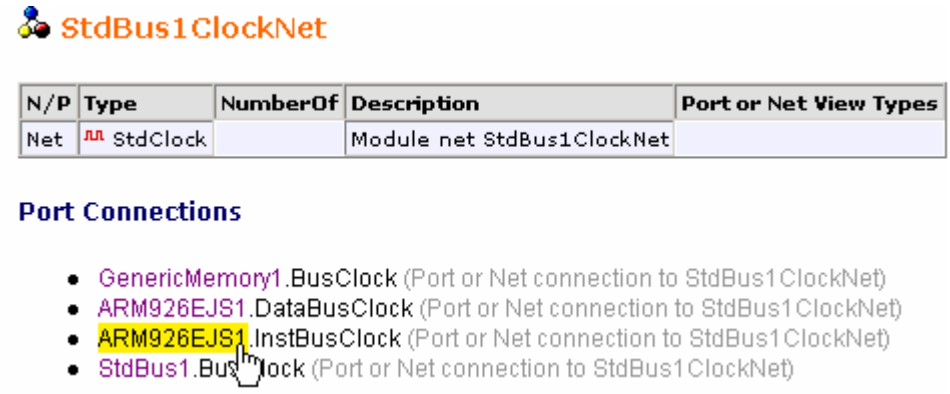
InstanceName	ModuleType	RequiredVersion	NumberOf	Description
VastGpStdLogic01	VastGpStdLogic0			
GenericMemory1	GenericMemory		NumberOfMemoryBlocks	This is an array of module instances, with the size of the array determined by the NumberOfMemoryBlocks parameter.
ARM926EJS1	ARM926EJS			
StdBus1	StdBus			

The Fabric Module Definition report displays a menu allowing you to browse to Properties, Parameters, Ports, Module Instances, Nets, Tasks and Interfaces. The Summary section contains concise tables of the various items. You can use links in the tables to drill down to a detailed view of each item. ModuleInstance ModuleType column entries link to the FmxReport for that module. Name column entries link to the detail entry for that name.

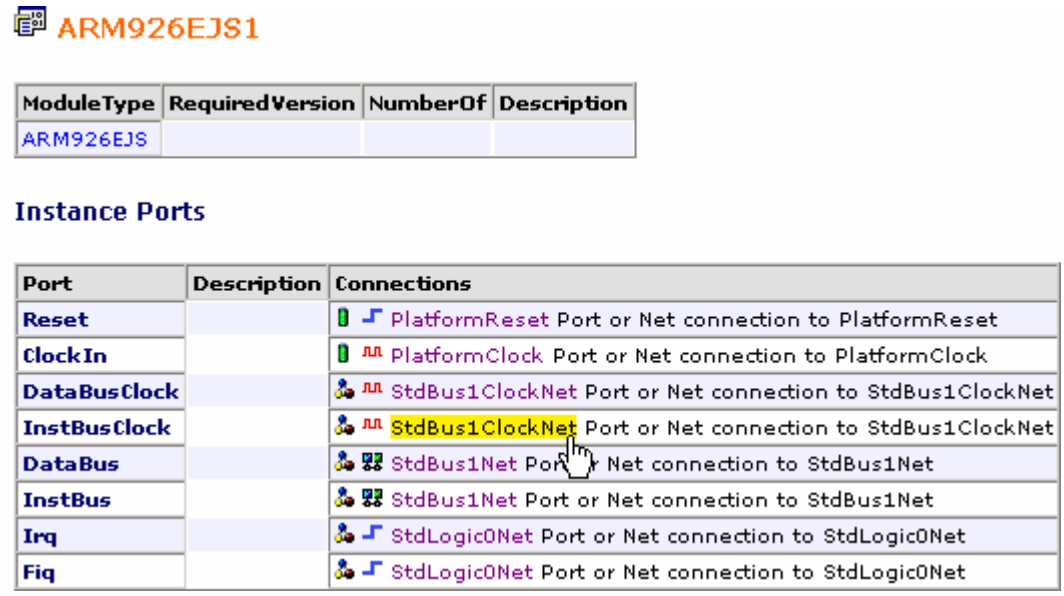
Nets

Name	Type	NumberOf	Description	Connections	Port or Net View Types
StdBus1ClockNet	StdClock		Module net StdBus1ClockNet	4	
StdBus1Net	StdBus		Module net StdBus1Net	4	
StdLogic0Net	StdLogic		Module net StdLogic0Net	3	

For example, click a link in the Name column of the Nets summary to see the detailed entry for that Net.



In the Net detail, you can view a list of all ports to which the net is connected. You can view a similar display in Port detail. Click on a module instance link in the Connections list to move to the detail for that Module Instance.



The Module Instance Detail lists Instance Ports and their connections. Icons indicate whether the connection is a net or a port and port or net type. Click on a link in the Connections column to move to the detail entry for the connected port or net.

Entries in the Module Type column link to the Fmx Report for the module from which the Module Instance is derived.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

As well as properties, Instance Ports and connections, Module Instance Detail entries show a table of Parameter Overrides.

GenericMemory1

Type	RequiredVersion	NumberOf	Class	SubClass	Description
GenericMemory		NumberOfMemoryBlocks	None		This is an array of module instances, with the size of the array determined by the NumberOfMemoryBlocks parameter.

Instance Ports

Port	Description	Connections
BusClock	Module port BusClock	StdBus1ClockNet
Reset	Module port Reset	PlatformReset
Bus	Module port Bus	StdBus1Net
Enable	Module port Enable	
MemoryPageSelect	Module port MemoryPageSelect	

Parameter Overrides

Name	Type	Value	Description
BurstReadBeatLimit	tWord32	0	Maximum beats in a read burst. Each beat = MemoryWidth of bytes. If 0 then no burst limit.
BurstWriteBeatLimit	tWord32	0	Maximum beats in a write burst. Each beat = MemoryWidth of bytes. If 0 then no burst limit.
EnablePowerOnDecode	tBoolean	1	Setting this Parameter to TRUE will allow this Memory Model to have its Decode available at Power On to facilitate Loading by VPM
MemorySize	tWord32	1 * M	Total Size in Bytes of the memory being attached to the Bus via this memory controller
MemoryWidth	tWord32	4	Width of the memory being attached to the Bus via this memory controller
Profiling	tAltList	Disabled	
StartAddress	tWord64	0	Base Address of the memory being attached to the bus via this memory controller
UseEnableInput	tBoolean	0	Setting this Parameter to TRUE will allow the Enable Pin to be Used to Enable/Disable this Memory
MemoryPaging			
NoOfMemoryPages	tWord32	1	Specifies the number of Memory Pages within the GP Memory Model
PageSize	tWord32	1 * M	Specifies the Page Size in Bytes if NoOfMemoryPages > 1. Otherwise Ignored
Timing			
FirstRead	tWord32	1	BusClock tick delay on Data Phase inserted for first read of MemoryWidth data of a burst
FirstWrite	tWord32	1	BusClock tick delay on Data Phase inserted for first write of MemoryWidth data of a burst
InitialRead	tWord32	0	BusClock tick delay on Data Phase inserted on initiating first read burst to memory
InitialWrite	tWord32	0	BusClock tick delay on Data Phase inserted on initiating first write burst to memory
NextRead	tWord32	1	BusClock tick delay on Data Phase inserted for each read of MemoryWidth data of a burst
NextWrite	tWord32	1	BusClock tick delay on Data Phase inserted for each write of MemoryWidth data of a burst

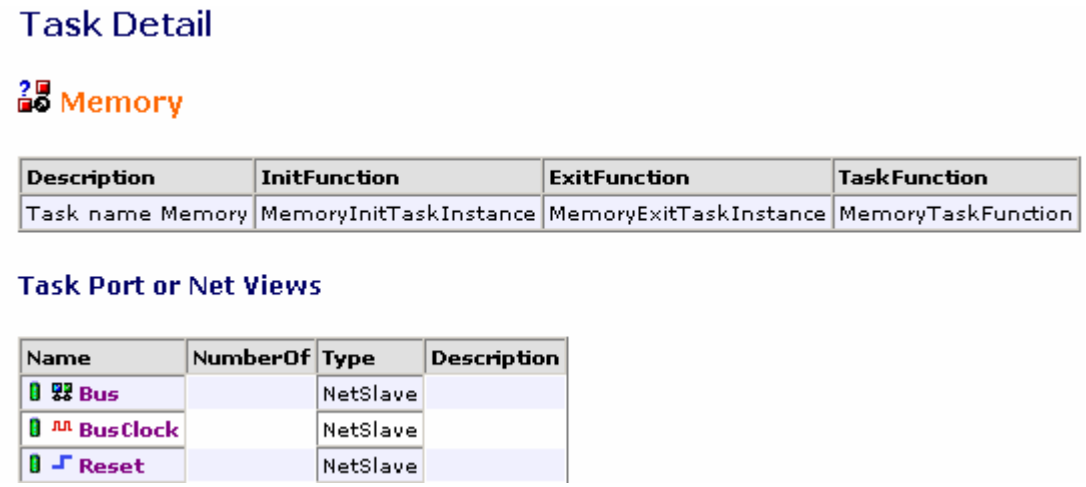
The module Ports and module Nets summaries show ports, properties, the number of connections and the port or net view types. If the connection count is zero, it is highlighted. In some cases a zero connection count may indicate an error.

Ports

Name	Type	NumberOf	Description	Connections	Port or Net View Types
Bus	StdBus		Module port Bus	0	NetSlave
BusClock	StdClock		Module port BusClock	0	NetSlave
Reset	StdLogic		Module port Reset	0	NetSlave

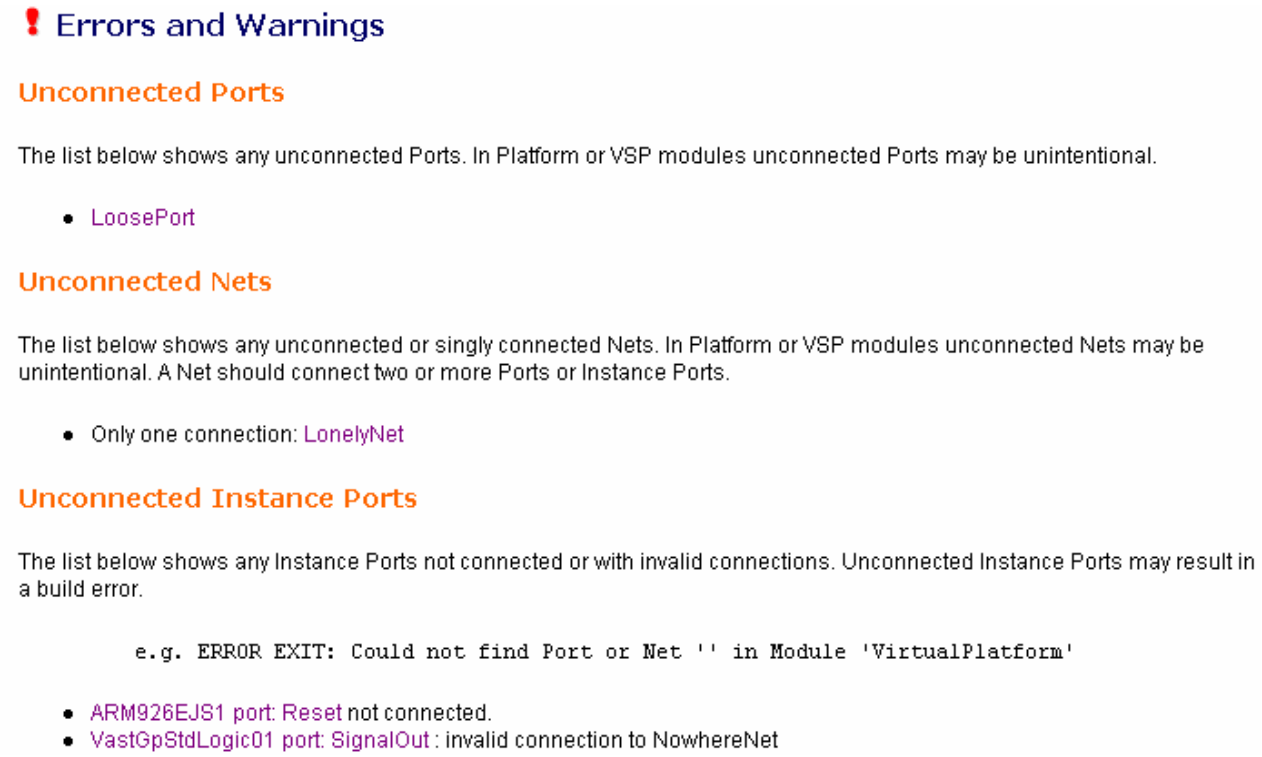
Click on the connections column entry to move to the connections list in the related Port or Net detail entry.

Click on a link in the Port or Net View Types column to move to the related Task detail.



Fmx Report Errors and Warnings

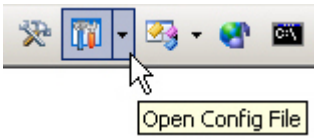
Click the Errors and Warnings link to see a list of errors and warnings. Not all issues reported are errors, depending on whether the module is a Platform, VSP or Peripheral Device. Warnings link to the detail for the related element.


Errors and Warnings
Unconnected Ports
The list below shows any unconnected Ports. In Platform or VSP modules unconnected Ports may be unintentional.
LoosePort
Unconnected Nets
The list below shows any unconnected or singly connected Nets. In Platform or VSP modules unconnected Nets may be unintentional. A Net should connect two or more Ports or Instance Ports.
Only one connection: LonelyNet
Unconnected Instance Ports
The list below shows any Instance Ports not connected or with invalid connections. Unconnected Instance Ports may result in a build error.
e.g. ERROR EXIT: Could not find Port or Net '' in Module 'VirtualPlatform'
ARM926EJS1 port: Reset not connected.
VastGpStdLogic01 port: SignalOut : invalid connection to NowhereNet

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Viewing a Prototype Configuration Report - Pcx Report

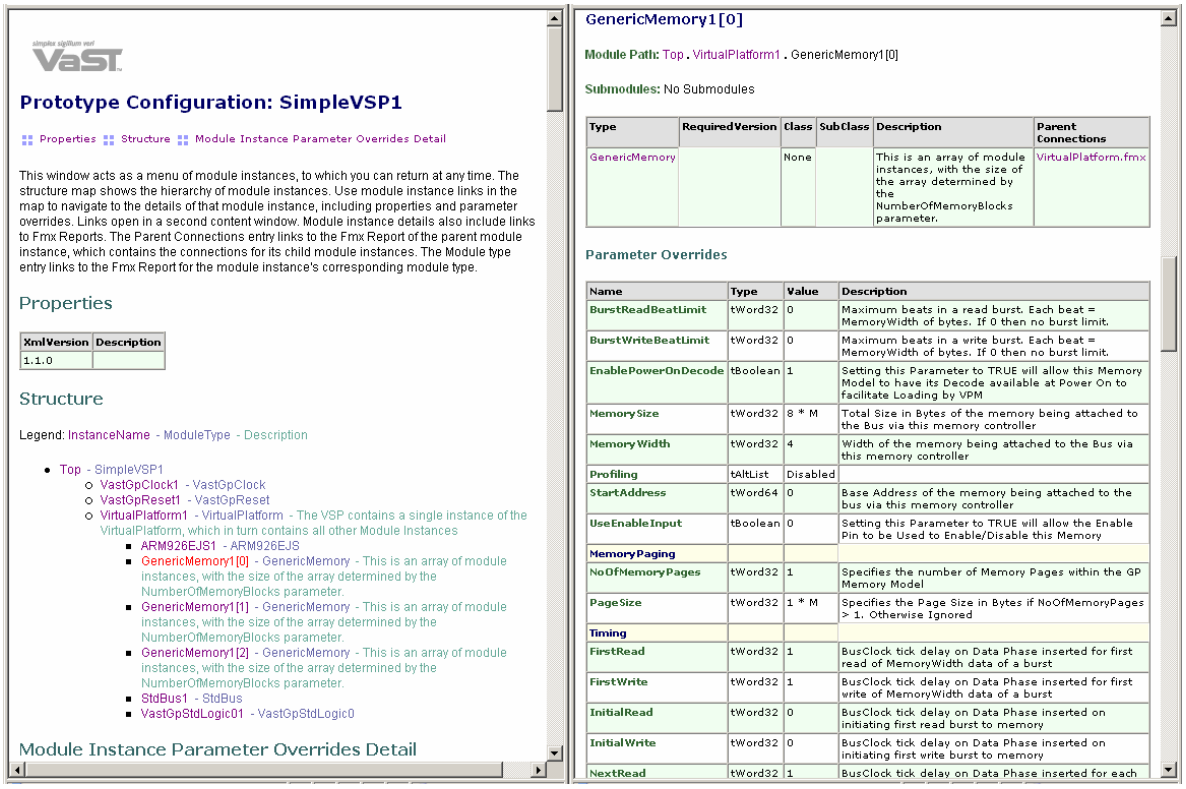
Open the Prototype Configuration (.pcx) file in the Document Window.



Click the Open Config File button down arrow and select the Prototype Configuration file from the list.

Select **Tools/Prototype Report** or click the Prototype Report button .

CoMET opens the report in the default XML browser.



Prototype Configuration: SimpleVSP1

Properties Structure Module Instance Parameter Overrides Detail

This window acts as a menu of module instances, to which you can return at any time. The structure map shows the hierarchy of module instances. Use module instance links in the map to navigate to the details of that module instance, including properties and parameter overrides. Links open in a second content window. Module instance details also include links to Fmx Reports. The Parent Connections entry links to the Fmx Report of the parent module instance, which contains the connections for its child module instances. The Module type entry links to the Fmx Report for the module instance's corresponding module type.

Properties

XmlVersion	Description
1.1.0	

Structure

Legend: InstanceName - ModuleType - Description

- Top - SimpleVSP1
  - VastGpClock1 - VastGpClock
  - VastGpReset1 - VastGpReset
  - VirtualPlatform1 - VirtualPlatform - The VSP contains a single instance of the VirtualPlatform, which in turn contains all other Module Instances
    - ARM926EJS1 - ARM926EJS
    - GenericMemory1[0] - GenericMemory - This is an array of module instances, with the size of the array determined by the NumberOfMemoryBlocks parameter.
    - GenericMemory1[1] - GenericMemory - This is an array of module instances, with the size of the array determined by the NumberOfMemoryBlocks parameter.
    - GenericMemory1[2] - GenericMemory - This is an array of module instances, with the size of the array determined by the NumberOfMemoryBlocks parameter.
    - StdBus1 - StdBus
    - VastGpStdLogic01 - VastGpStdLogic0

Module Instance Parameter Overrides Detail

GenericMemory1[0]

Module Path: Top . VirtualPlatform1 . GenericMemory1[0]

Submodules: No Submodules

Type	RequiredVersion	Class	SubClass	Description	Parent Connections
GenericMemory		None		This is an array of module instances, with the size of the array determined by the NumberOfMemoryBlocks parameter.	VirtualPlatform.fmx

Parameter Overrides

Name	Type	Value	Description
BurstReadBeatLimit	tWord32	0	Maximum beats in a read burst. Each beat = MemoryWidth of bytes. If 0 then no burst limit.
BurstWriteBeatLimit	tWord32	0	Maximum beats in a write burst. Each beat = MemoryWidth of bytes. If 0 then no burst limit.
EnablePowerOnDecode	tBoolean	1	Setting this Parameter to TRUE will allow this Memory Model to have its Decode available at Power On to facilitate Loading by VPM
MemorySize	tWord32	8 * M	Total Size in Bytes of the memory being attached to the Bus via this memory controller
MemoryWidth	tWord32	4	Width of the memory being attached to the Bus via this memory controller
Profiling	tAltList	Disabled	
StartAddress	tWord64	0	Base Address of the memory being attached to the bus via this memory controller
UseEnableInput	tBoolean	0	Setting this Parameter to TRUE will allow the Enable Pin to be Used to Enable/Disable this Memory
MemoryPaging			
NoOfMemoryPages	tWord32	1	Specifies the number of Memory Pages within the GP Memory Model
PageSize	tWord32	1 * M	Specifies the Page Size in Bytes if NoOfMemoryPages > 1. Otherwise Ignored
Timing			
FirstRead	tWord32	1	BusClock tick delay on Data Phase inserted for first read of MemoryWidth data of a burst
FirstWrite	tWord32	1	BusClock tick delay on Data Phase inserted for first write of MemoryWidth data of a burst
InitialRead	tWord32	0	BusClock tick delay on Data Phase inserted on initiating first read burst to memory
InitialWrite	tWord32	0	BusClock tick delay on Data Phase inserted on initiating first write burst to memory
NextRead	tWord32	1	BusClock tick delay on Data Phase inserted for each

The report shows the prototype properties, and a tree view of the prototype structure. Use the module instance links in the Structure map to navigate to module instance detail entries.

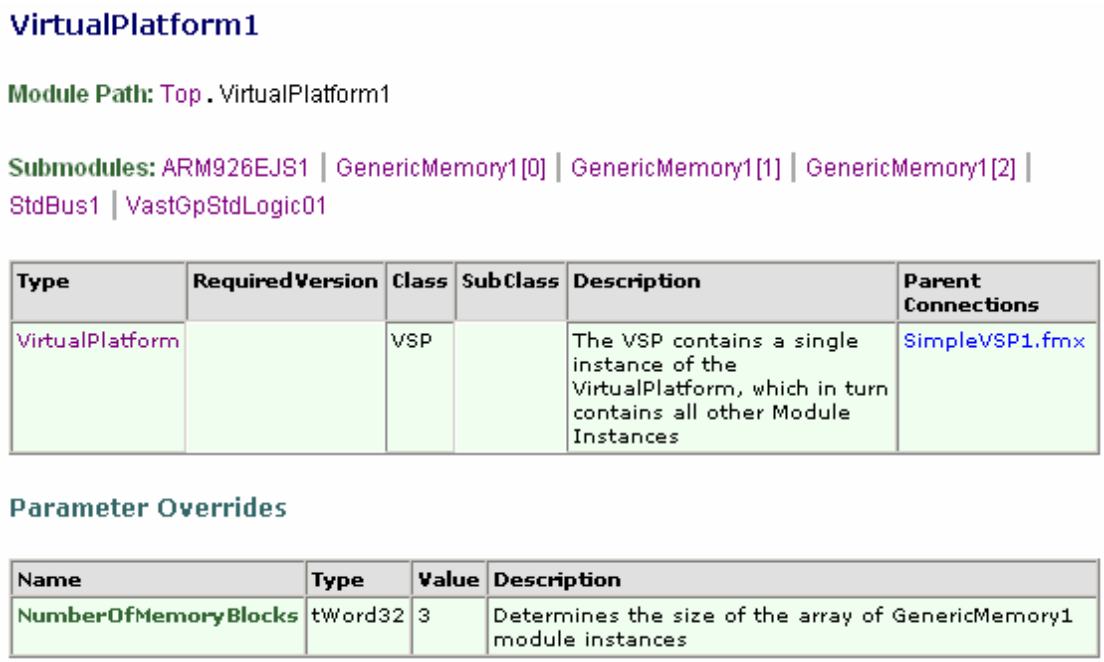
The Pcx Report effectively has two windows. This helps in navigating complex hierarchies. The first window opens when you first open the report. When you click on a link in the first window, a second window opens. Position the two windows side by side.

Now you can use the first window Structure list as a menu to select module instances and other items in the second window.

If you prefer to use a single window, close the first window after opening the second window.

May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

To return to the Structure map, move to the top of the file. The hot key Ctrl-Home performs this function in Microsoft Internet Explorer and Mozilla.



VirtualPlatform1

Module Path: Top . VirtualPlatform1

Submodules: ARM926EJS1 | GenericMemory1[0] | GenericMemory1[1] | GenericMemory1[2] | StdBus1 | VastGpStdLogic01

Type	RequiredVersion	Class	SubClass	Description	Parent Connections
VirtualPlatform		VSP		The VSP contains a single instance of the VirtualPlatform, which in turn contains all other Module Instances	SimpleVSP1.fmx

Parameter Overrides

Name	Type	Value	Description
NumberOfMemoryBlocks	tWord32	3	Determines the size of the array of GenericMemory1 module instances

Detail entries show a Module Path, Submodule list and module instance properties. All Parameter Overrides are shown in a table.

Use links in the Module Path list to navigate to higher levels of the hierarchy. Use links in the Submodules list to navigate to submodules.

The ModuleType entry links to the Fmx file for the Module corresponding to this module instance.

Viewing All Connections in a Virtual System Prototype

The Module Instance Parent Connections entry links to the detail entry for this Module Instance in its parent module Fmx Report, which shows all the connections for this Module Instance. By navigating to module instances and viewing their parent connections, you can use the Pcx Report to view every module instance in the Virtual System Prototype, and its connections.

To navigate around the hierarchy, click links in the Module Path list, the Submodules list, or return to the top of the file (Ctrl-Home in IE, Mozilla) to use the Structure tree map. Click ModuleType links and Parent Connections links to view Fmx Reports By using the first window as a menu you can easily return to base.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

GenericMemory1[0]

Module Path: Top . VirtualPlatform1 . GenericMemory1[0]

Submodules: No Submodules

Type	RequiredVersion	Class	SubClass	Description	Parent Connections
GenericMemory		None		This is an array of module instances, with the size of the array determined by the NumberOfMemoryBlocks parameter.	VirtualPlatform.fmx

Parameter Overrides

Name	Type	Value	Description
BurstReadBeatLimit	tWord32	0	Maximum beats in a read burst. Each beat = MemoryWidth of bytes. If 0 then no burst limit.
BurstWriteBeatLimit	tWord32	0	Maximum beats in a write burst. Each beat = MemoryWidth of bytes. If 0 then no burst limit.
EnablePowerOnDecode	tBoolean	1	Setting this Parameter to TRUE will allow this Memory Model to have its Decode available at Power On to facilitate Loading by VPM
MemorySize	tWord32	8 * M	Total Size in Bytes of the memory being attached to the Bus via this memory controller
MemoryWidth	tWord32	4	Width of the memory being attached to the Bus via this memory controller
Profiling	tAltList	Disabled	
StartAddress	tWord64	0	Base Address of the memory being attached to the bus via this memory controller
UseEnableInput	tBoolean	0	Setting this Parameter to TRUE will allow the Enable Pin to be Used to Enable/Disable this Memory
MemoryPaging			
NoOfMemoryPages	tWord32	1	Specifies the number of Memory Pages within the GP Memory Model
PageSize	tWord32	1 * M	Specifies the Page Size in Bytes if NoOfMemoryPages > 1. Otherwise Ignored
Timing			
FirstRead	tWord32	1	BusClock tick delay on Data Phase inserted for first read of MemoryWidth data of a burst
FirstWrite	tWord32	1	BusClock tick delay on Data Phase inserted for first write of MemoryWidth data of a burst
InitialRead	tWord32	0	BusClock tick delay on Data Phase inserted on initiating first read burst to memory
InitialWrite	tWord32	0	BusClock tick delay on Data Phase inserted on initiating first write burst to memory
NextRead	tWord32	1	BusClock tick delay on Data Phase inserted for each read of MemoryWidth data of a burst
NextWrite	tWord32	1	BusClock tick delay on Data Phase inserted for each write of MemoryWidth data of a burst

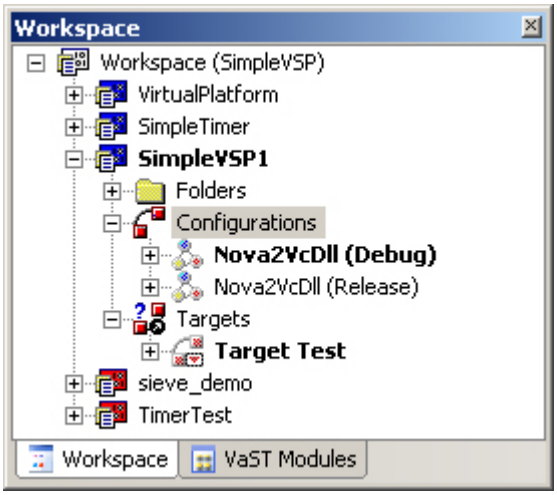
Where applicable, Module Instance detail entries show a table of parameter overrides.

May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Configuring Build and Runtime Options

A project build configuration specifies details of how the project is to be built, such as include paths, assembler, compiler and linker options, and runtime options.

For each new project, CoMET automatically assigns two types of build configuration: a debug configuration and a release configuration. You can modify each build configuration using different characteristics, and build and test the module for each specific build configuration. You can also add your own build configurations.

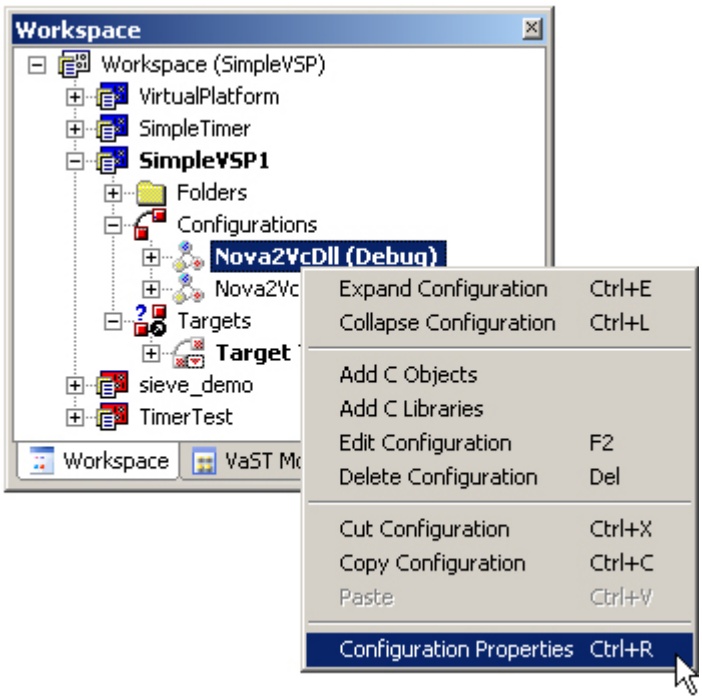


Setting the Build Configuration

You can set different parameters for each build configuration. The build configuration parameters vary for different project types (Fabric, VBus, Metrix, Target or General) and module classes.

To set parameters for the build configuration:

- Right mouse-click on the build configuration icon or name in the workspace tree.



- Select Configuration Properties

or

- Double-click on the configuration icon

CoMET displays the Project dialog, Build Configuration tab, with attributes appropriate to the Build Configuration type.

Build Configuration Types

The build configuration type depends on:

- **Simulator**: Nova2:Nova CIF or Nova, or other
- **Compiler**: Vc:Visual C or Gcc
- **Type of output executable**: App:application or Dll:Dynamic Linking Library

The build configuration type name is created by concatenating these specifiers: e.g. `Nova2VcDll`.

Common Build Configuration Properties

All projects have the following fields in common:

Type: Build Configuration Type. See *Build Configuration Types, page 110*.

Mode: Debug or Release

Path: The destination folder for the output files.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

NOTE:	If you create a Build Configuration using the same Path as used for a previous Build Configuration, you must choose the **Rebuild All** option on the next build, to ensure that all output files are replaced with newly built files.

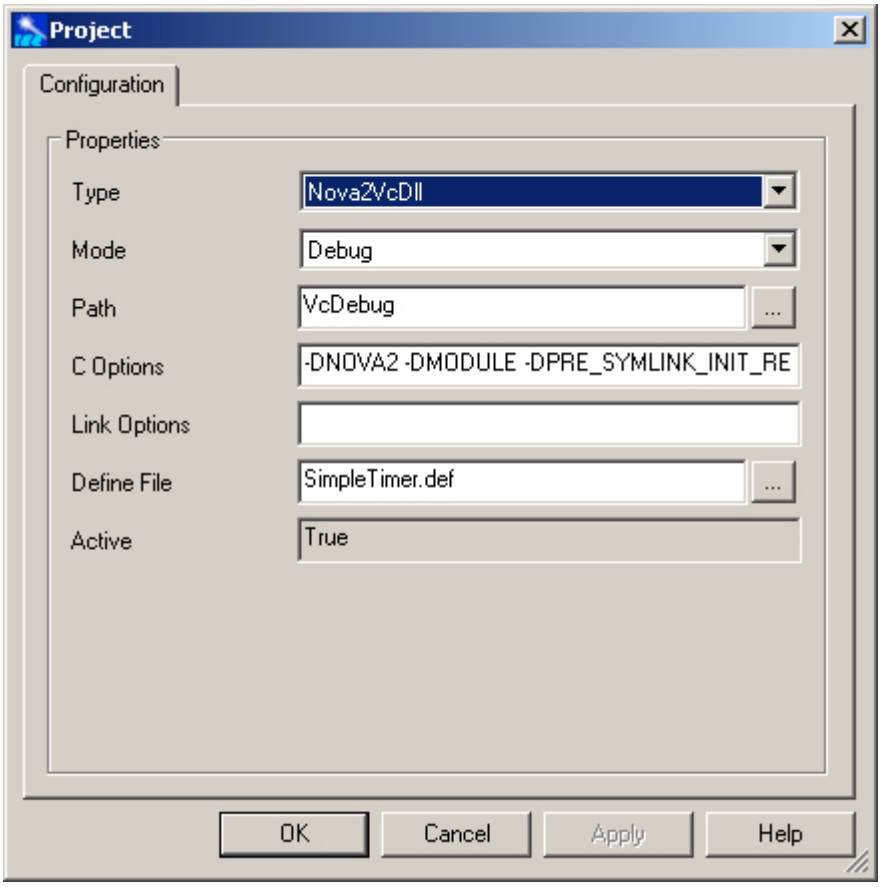
C Options: C compiler command line options

Link Options: Linker command line options

Define File: Certain project definitions

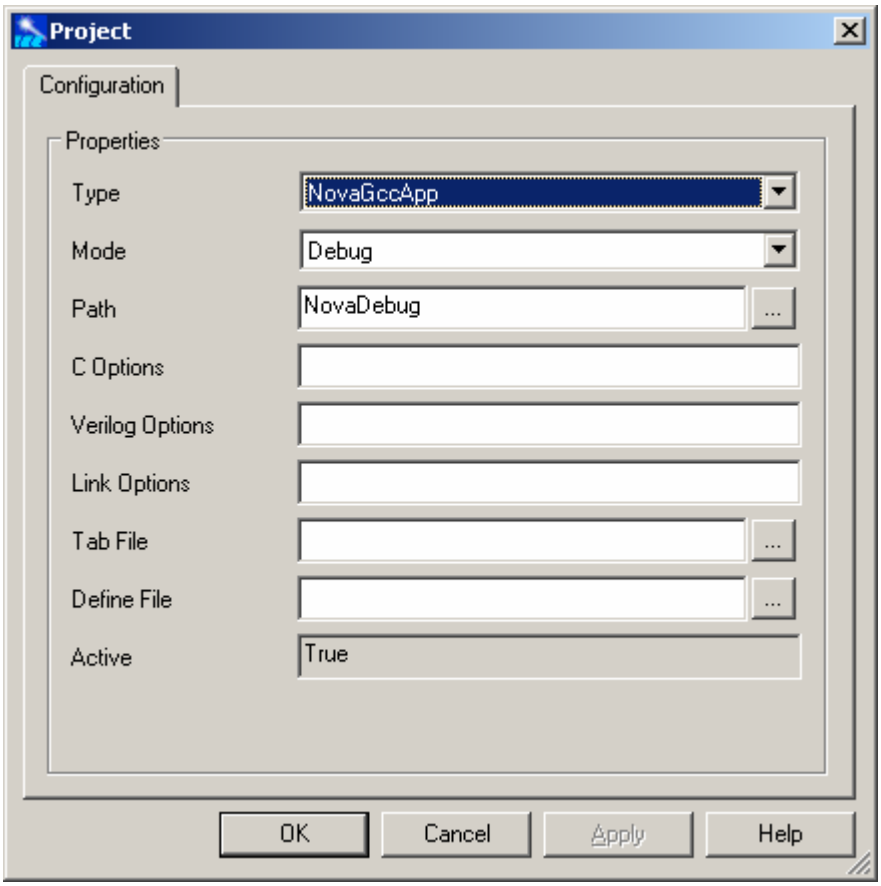
Active: Active status, True or False - the Build command builds the project using the Build Configuration with Active attribute set to True. When the value is False, a drop down list allows it to be changed to True.

CIF Project Build Configuration Properties



A CIF Project Build Configuration has the common set of properties. See *Common Build Configuration Properties, page 110*.

VBus Standalone Build Configuration Parameters



See *Common Build Configuration Properties, page 110*

Verilog Options: Command line options for the Verilog simulator or compiler

Tab File: Specifies the PLI (Program Language Interface) Table (.tab) required for this build. See *Tab files are no longer required, page 112*.

Tab files are no longer required

The PLI (Program Language Interface) Table (.tab) file was used in conjunction with Verilog or NOVA modules using Behavioral C PLI functions.

The .tab file contained declarations of PLI functions. This is not required for Fabric modules.

For non-Fabric projects that use Behavioral C PLI functions there is an alternative approach to using .tab files. If you have Behavioral C projects that use .tab files, you may continue to use them, or you can update the projects to operate without .tab files.

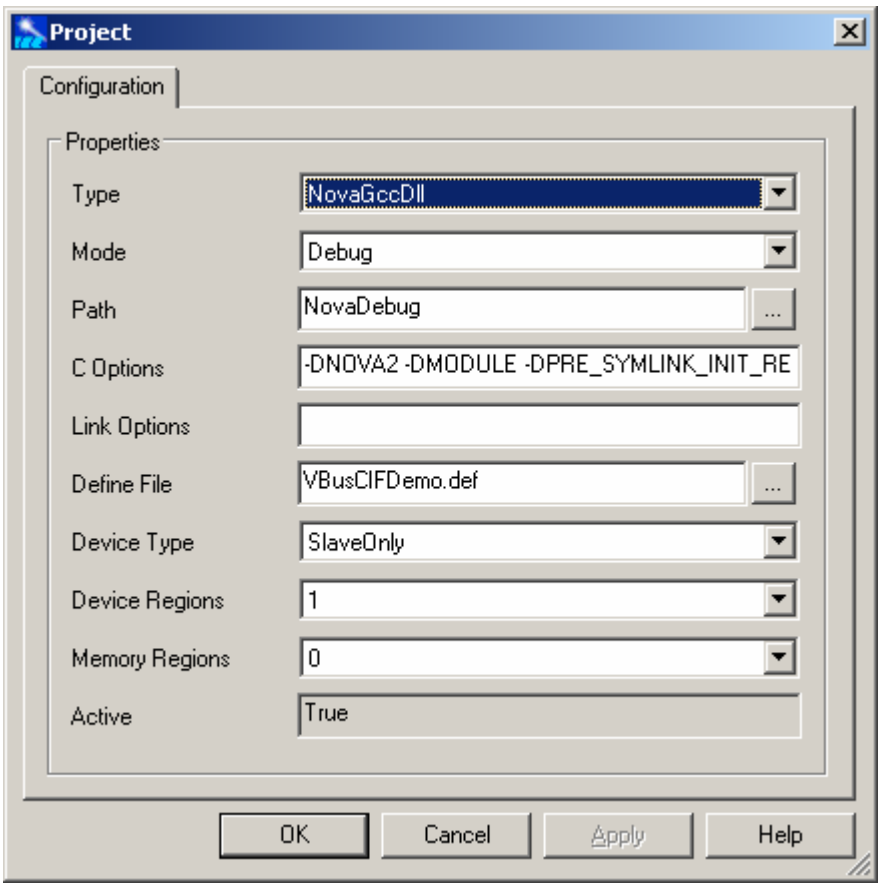
To update such projects to use no tab file:

- Remove the $vpa_ prefix from all user tasks
- Change the function type of all user tasks from void to tAmpiUserTask
- Delete the entry in the .tab file field from the HDL tab of the project settings dialogue box
- Delete the .tab file, as it is no longer required.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Alternatively, to use .tab files, specify the required .tab file in the VBus project dialog Tab File field. If you leave the Tab file field blank, CoMET assumes there is no .tab file and operates in the No Tab File mode.

VBus CIF Build Configuration Properties



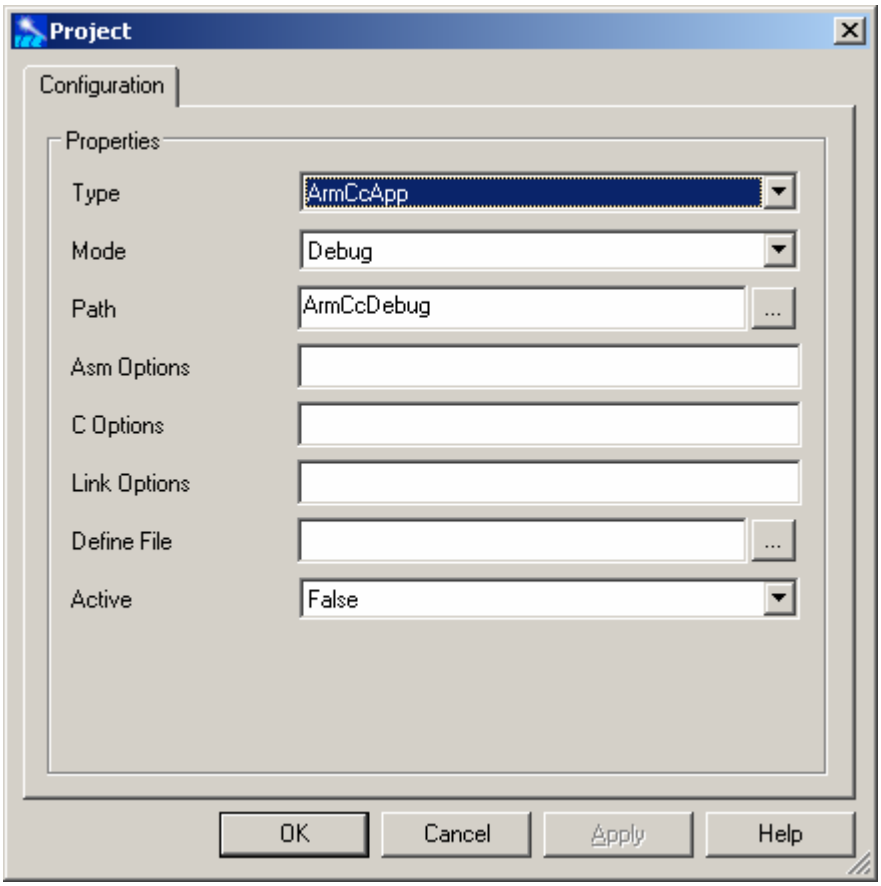
See *Common Build Configuration Properties, page 110*

Device Type: Allowable Values: `SlaveOnly` (Default), `MasterSlave`, `NonBus`

Device Regions: The number of device decoding regions. Default `1`

Memory Regions: The number of memory decoding regions. Default: `0`

Target Project Build Configuration Properties



See *Common Build Configuration Properties, page 110*

Asm Options: Assembler command line options

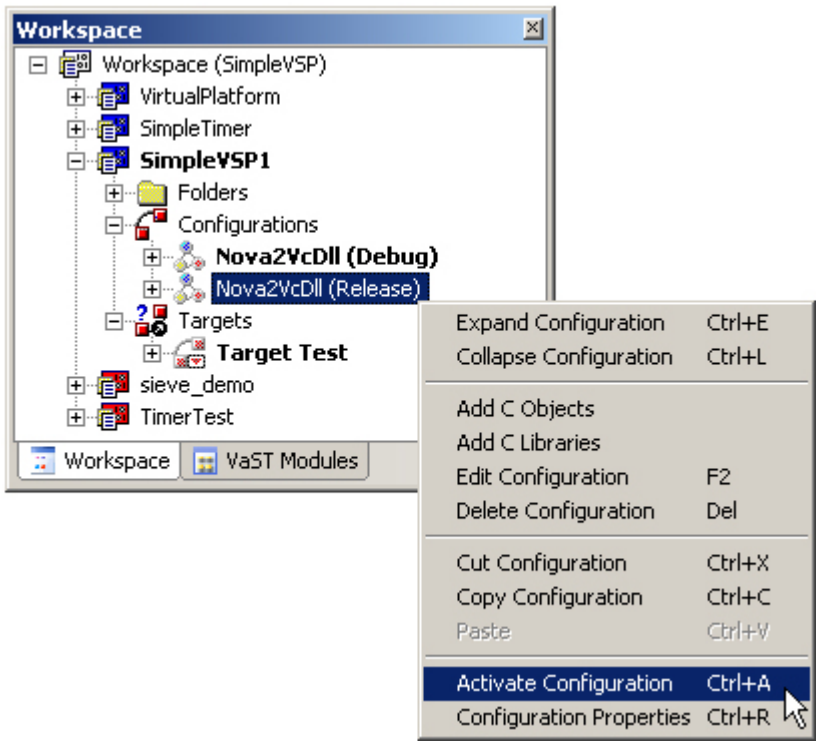
Setting the Active Build Configuration

A project (module) may contain several build configurations. When you build the project, it is the active build configuration that is built.

To set the active configuration:

- Right mouse-click on the configuration icon

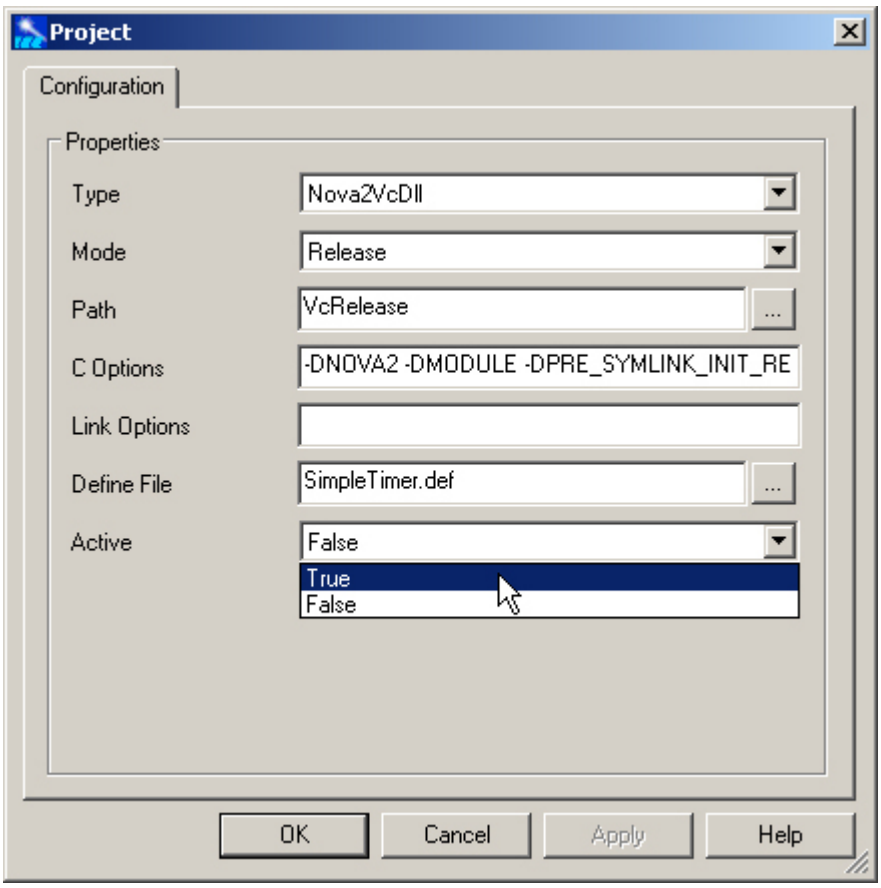
VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.



- Select **Activate Configuration**

or

- Double-click on the configuration icon



- Set the **Active** field to **True** and click the **OK** button.

NOTE: When CoMET builds a project in debug mode, debug information that allows you to debug the behavioral code is stored in the executable. Debug information adds a significant overhead, which may degrade performance. To run the model at maximum performance, you should configure and build a model in release mode.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Setting the Runtime Options

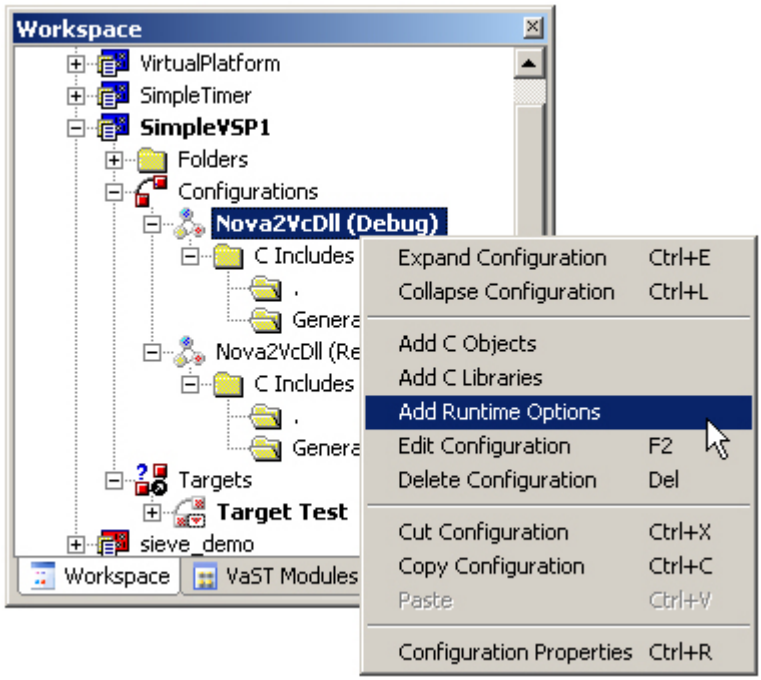
The build configuration of the top module supports a set of runtime options. Runtime Option properties vary for different build configuration types (See *Build Configuration Types, page 110)*.

Adding Runtime Option Properties

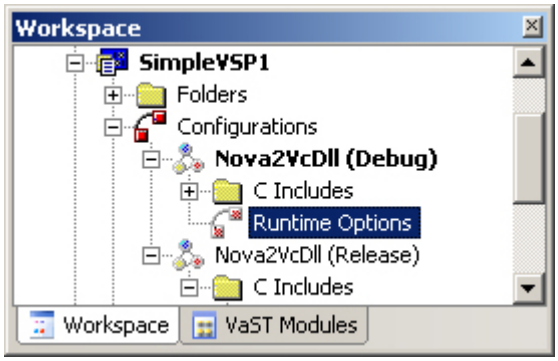
To gain access to view and change Runtime Options you need to create a Runtime Options node in the Workspace window, under the Configuration node. CoMET creates a Runtime Options node automatically only for top level Fabric projects and VBus projects. You can manually add a Runtime Options node.

To add runtime options:

- Right click on the appropriate build Configuration node
- Choose **Add Runtime Options** from the context menu.



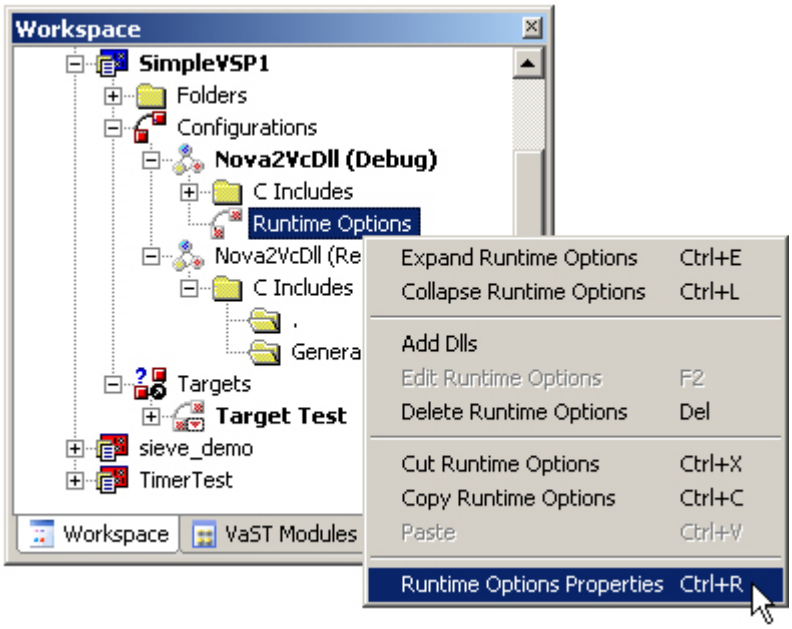
CoMET creates a new Runtime Options node under the build Configuration node.



Viewing and Setting Runtime Options Properties

To set the runtime options for the top module:

- Right mouse-click on **Runtime Options**



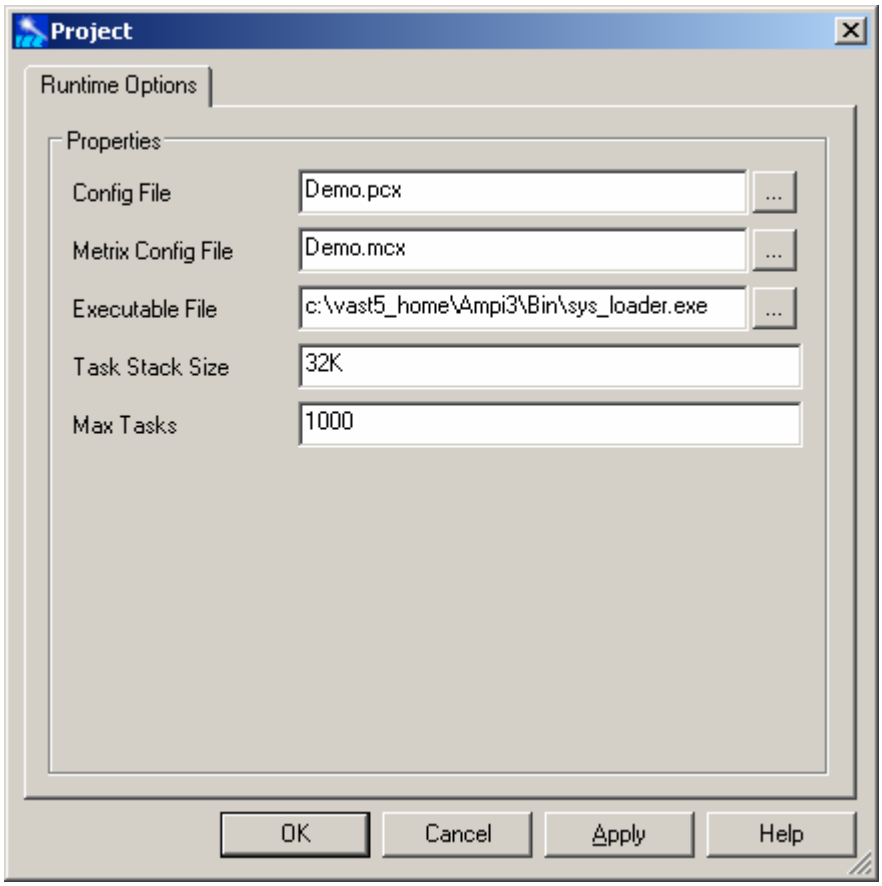
- Select **Runtime Options Properties**

or

- Double-click on **Runtime Options**

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly
authorized in a written agreement with VaST.

Fabric Project Runtime Options



Config File: The Prototype Configuration file for the project. See *Modifying Parameter Values in the Prototype* Configuration File*, page 121*.

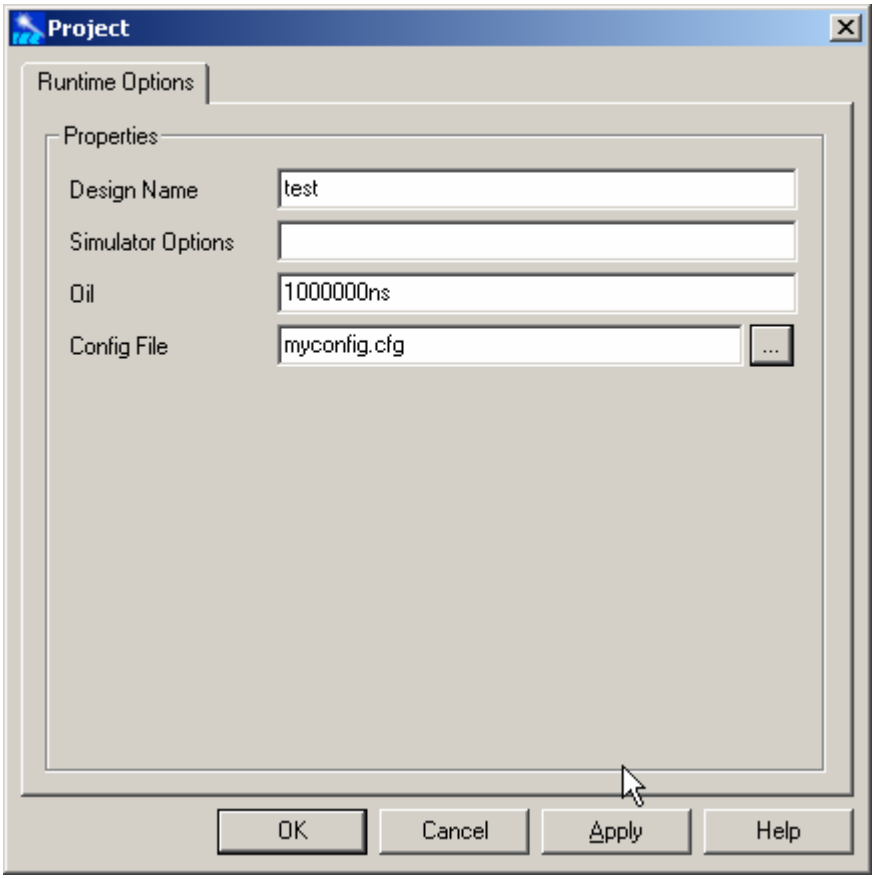
Metrix Config File: The path of the associated Metrix Configuration (.mcx) file. See the *Metrix Configuration File User Guide*.

Executable File: This specifies the file that calls the Fabric .dll.

Task Stack Size: Default 32K. May be increased for large projects.

Max Tasks: Default 1000. May be increased for large projects.

VBus Module Runtime Options Properties



Design Name: Verilog design name

Simulator Options: These options are passed to the hardware simulator. See *Supported simulation engines, page 120.*

Oil: Optimum Interrupt Latency - This setting controls the maximum time difference allowed during the simulation between the software and hardware components. Higher OIL means better simulation performance but possibly late response to interrupts. Units are nanoseconds (ns)

Config File: The Architecture Configuration File for this project.

Supported simulation engines

This version of CoMET supports only Nova1 for VBus and Fabric projects, Nova2 (Nova_CIF) for Fabric CIF projects and Mentor Graphics ModelSim for Fabric VHDL projects. Other hardware simulators such as FinSim and NCSIM maybe supported in a later version.

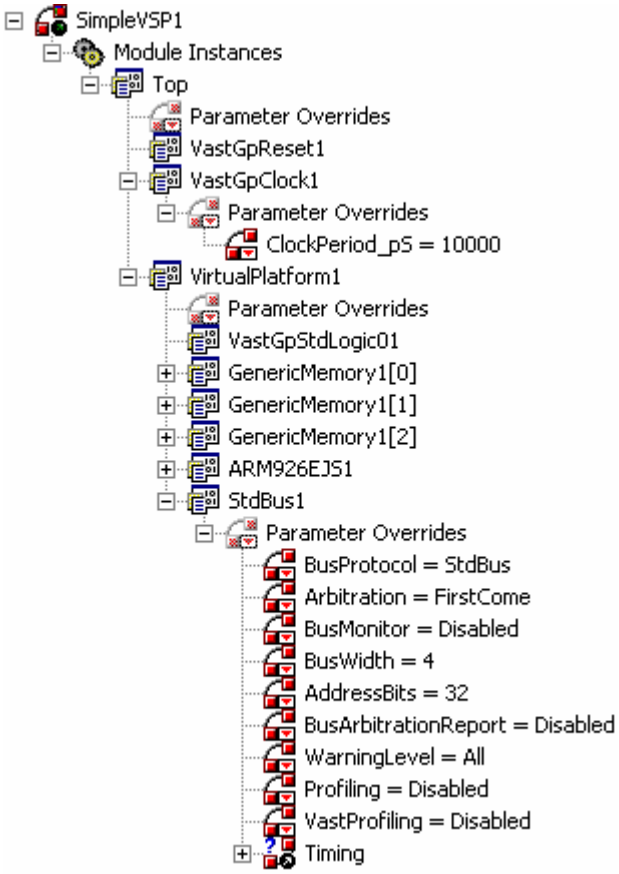
VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Modifying Parameter Values in the Prototype Configuration File

A VSP (Virtual System Platform) consists of a hierarchy of module instances, each of which may have Parameter Override elements. For each module instance, the corresponding module may itself contain module instances. This module instance hierarchy can be represented as a tree structure.

The Prototype Configuration (pcx) file provides access to Parameter Override elements, corresponding to Parameter elements for each module instance in the tree. The pcx file is evaluated at simulation time.

For details on the Parameter Override element, see *Parameter Override, page 89*. For details on the Module Instance element, see *Module Instance, page 84*.



The .pcx file above is arranged according to the hierarchy of module instances. The SimpleVSP1 module contains a VirtualPlatform1 module instance, which in turn contains Module Instances at a lower level, with Parameter Overrides for all Parameters of the corresponding modules.

To create a new configuration file:

- Choose **Workspace/Update Platform**

May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

A .pcx file is also created on a successful build.

To open the configuration file:

- From the toolbar, click the Open Config button:

Pcx Update - Update Platform

After changes to an .fmx file, the project .dll and the pcx file must be updated. The pcx file is created or updated on build. You can also create or update the Pcx file at any time without building the .dll.

To update the configuration file:

- From the Workspace menu, select **Update Platform.**

CoMET updates the .pcx to contain Override Parameters corresponding to all module instance Parameters in the VSP.

Update Platform determines the fmx hierarchy by examining the fmx files from the top down. It creates a corresponding Module Instance in the pcx tree. It then determines the Parameter elements for each Module Instance and populates the corresponding pcx Module Instance with corresponding Parameter Overrides. See *Fmx Update, page 72*.

When the .pcx is open in the Document window, you can edit it in XML Tree view (shown above) or XML Table parameter view. See *Parameter Views, page 98*.

You can view a comprehensive report on the .pcx file. See *Viewing a Prototype Configuration Report, page 105*.

Excluding Build-Time Parameters from Prototype Configuration Files

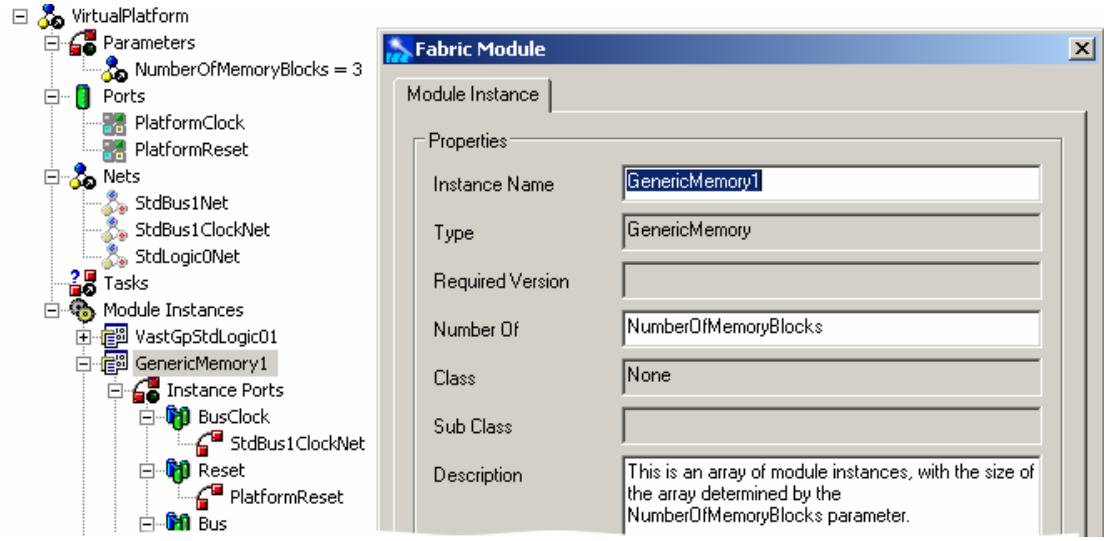
Pcx files are generated from .fmx files. It is not possible to make a change to a .pcx file which results in a change in an fmx file, that is, a change in the structure of the VSP.

Pcx files are intended for runtime Parameter Overrides. Pcx file Parameter Overrides are not evaluated until simulation time.

Fmx files may contain parameters which are not runtime parameters, but which affect VSP structure. These are build-time parameters, and should not appear in the pcx file.

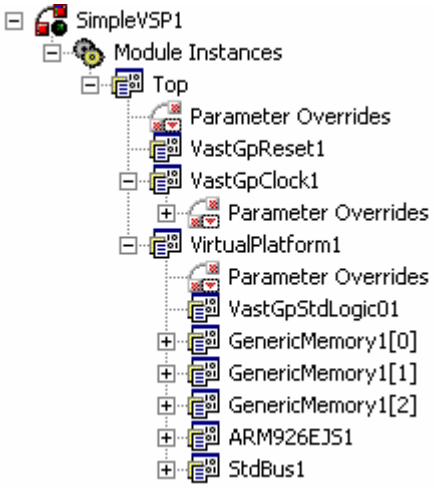
For example, it is common to use a platform level Parameter to specify the size of a Generic Memory Module Instance array. In the example below the Parameter is called NumberOfMemoryBlocks. This makes it easy to vary the number of memory blocks in a platform. You should vary this parameter by changing the value of the NumberOfMemoryBlocks Parameter Override in the platform module instance in the VSP.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.



The platform NumberOfMemoryBlocks parameter determines platform and VSP structure

You can then regenerate the .pcx file by rebuilding the VSP project. Module Instance Parameter Overrides appear in the pcx for each member of the array.



The generated pcx file shows a set of Module Instance Parameter Overrides for each member of the GenericMemory1 array.

Such a build-time parameter should not appear in the Pcx file. You can control whether a Parameter Override appears in the Pcx file using the Parameter Pcx Access attribute. Set Pcx Access to Private to prevent generation of a Pcx Parameter Override. See *Parameter, page 78*.

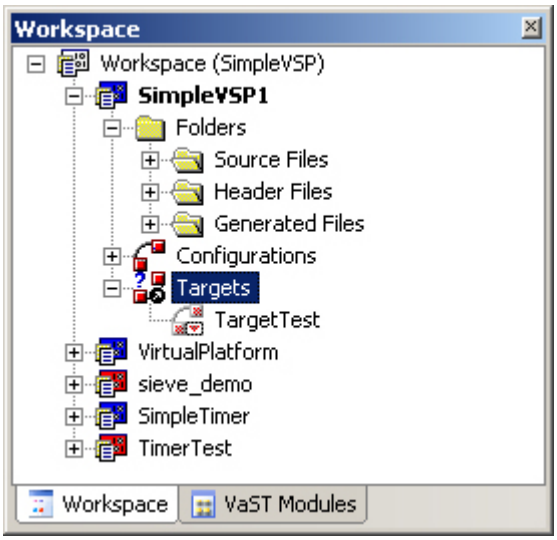
Note that the above pcx file does not contain a Parameter Override for the NumberOfMemoryBlocks parameter, as the NumberOfMemoryBlocks Parameter Pcx Access attribute is set to Private.

Setting the Target Executable

If the module includes a Virtual Processor Model instance, CoMET requires target code to run the simulation. When you create a module, CoMET automatically adds a target setting to contain target properties, such as the type of executable and the location of the executable image.

You can load a target image into any module that has been built with a target image loader interface. These include VPMs and some other modules such as memory devices.

Note that you can target-debug only target images loaded into VPM instances.



To add a target:

- Right click on **Targets** and choose **Add Target** from the context menu.

To delete a target:

- Right click on the target icon or name and choose **Delete Target** from the context menu.

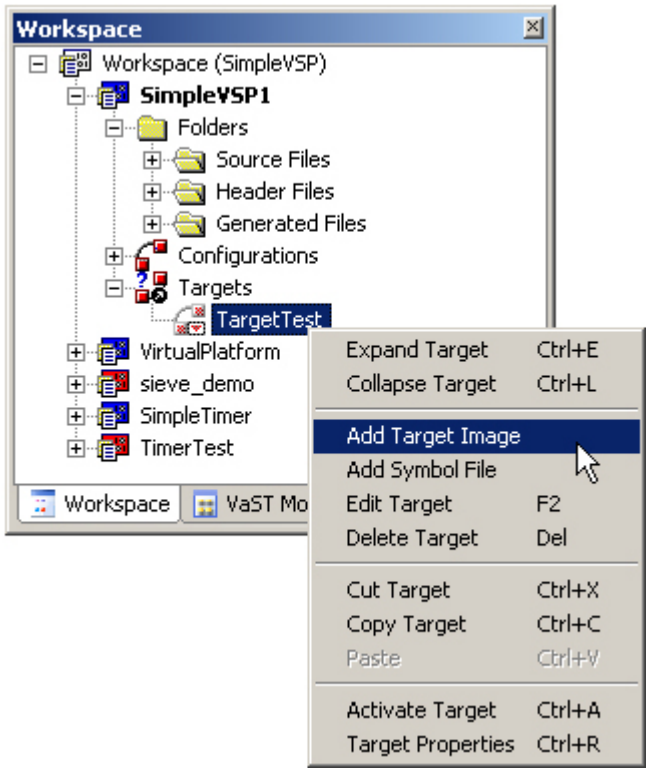
To change the name of the target or its Active status:

- Right click on the target icon or name and choose **Target Properties** from the context menu.

To add target information:

- Right mouse-click on the target icon or name**.**

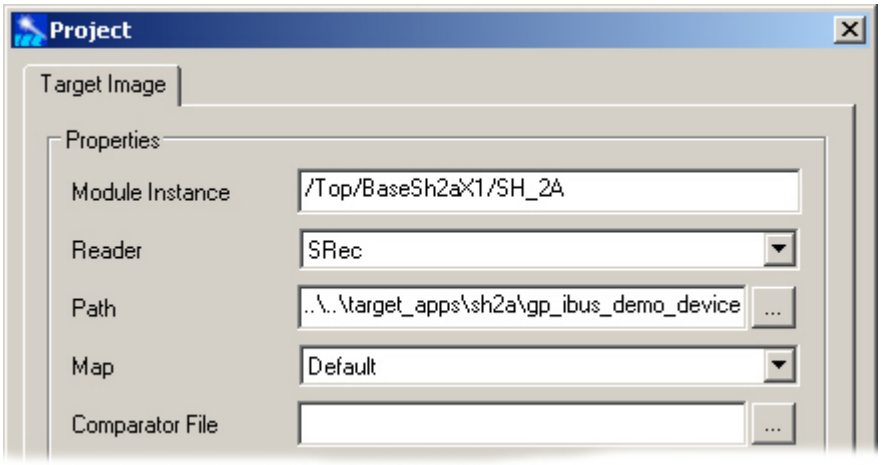
VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.



- Select **Add Target Image**

CoMET displays the Project dialog, Target Image Properties tab.

Target Image Properties



- In the Target Image dialog,

Module Instance:

- Specify the module into which the the target is to be loaded. CoMET provides a list of all available module instances with target image loader interfaces

May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Reader: Specify the binary format of the target image:

- Elf
- Coff
- SRec
- AbsToSrec
- IHex
- Default

Path:

- Browse and locate the image file

Map: Select the map to be used with this image

- `Default`: maps byte addressing to byte addressing (no translation)
- `TeakLiteCode`: maps word addressing to byte addressing (multiplies by 2)
- `TeakLiteData`: maps word addressing to byte addressing (multiplies by 2 and sets MSB to 1)
- `WinCE`: select this option when loading a binary image built by Microsoft Platform Builder for WinCE.

Comparator File:

- Specify the Target Image Comparator SREC file, if any, used for verification. There are restrictions on the Comparator File Format. See *Restrictions on the Target Image Comparator SREC format, page 127*. This field is optional.

- Click **OK**

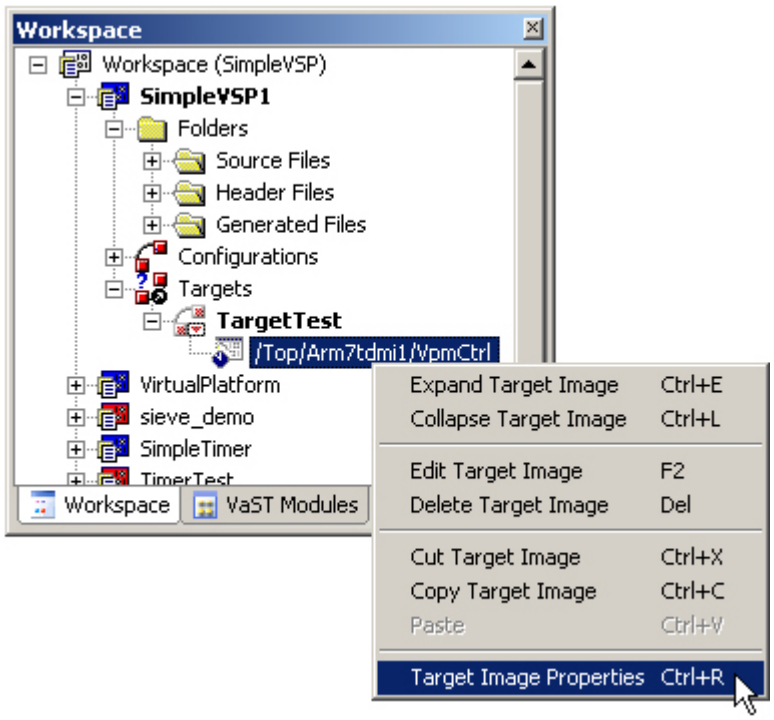
To alter Target Image properties at a later stage:

- Right click on the target image icon and select **Target Image Properties** from the context menu.

or

- Double click on the target image icon.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.



Restrictions on the Target Image Comparator SREC format

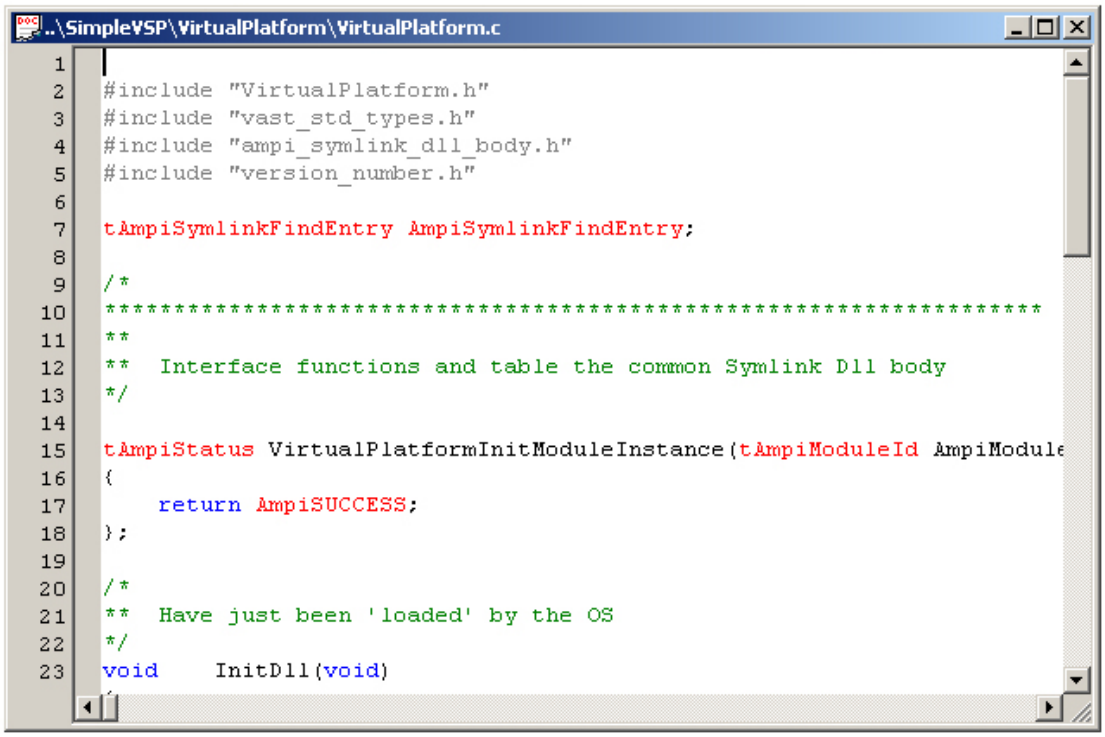
The SREC file format used in CoMET is a restricted version of the standard format. Restrictions are as follows:

- S records must be sorted in ascending order by Address field.
- Records must not overlap, i.e. No record may contain an Address that falls within the address range (start: Address, end: Address+Length-1) of another record.
- Any memory that does not fall within the address range of a record is assumed to contain zero.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Using the Source Editor

With the text editor provided by CoMET, you can edit the module source files (module description files and C source code files).

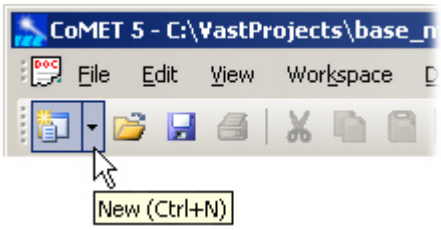


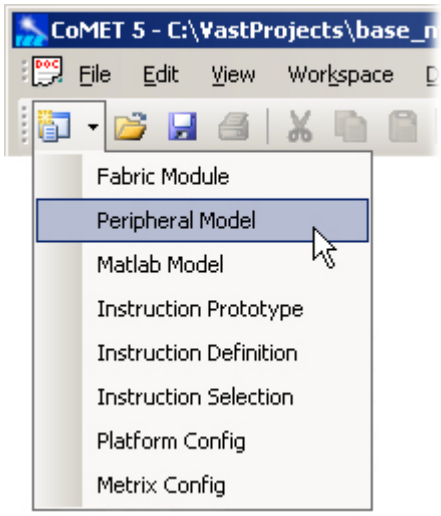
CoMET Code Editor Window

The module editor uses standard file and editing functions (**File** and **Edit** menus) for creating new files, opening existing files, searching, copying and pasting text, and so on. You can also use tools from the tool bar for commonly used commands, and standard keyboard shortcuts.

To create a new file:

- From the **Tool Bar**, click the down arrow next to **New File** button





- Select the type of file you wish to create from the drop down menu.

NOTE: Clicking the New File button will automatically create a new text file.

To open an existing file:

- From the **File** menu, select **Open File**
- Navigate to the directory in which the source file is stored
- Select the file name
- Click on the **Open** button.

or

- Double click on a source file in the Workspace Source Files or Header Files folders.

NOTE: Creating a new file or opening an existing file does not automatically add the file into the project. You must also add a reference to the file in the Project (.pjx) file, using the Workspace window interface, so that it appears in the Source Files or Header Files folder. See *Adding a Source File to a Project, page 55*.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Highlighting Keywords

By default, the module editor highlights certain keywords in the body of a source file, for example to separate the syntax of the language from module variables, etc. You can modify the list of keywords or add new keywords, for example to add user functions or other control syntax to the keyword list. The keywords are contained in four files:

- `%VAST5_HOME%/Config/verilogkeywords.txt` (for Nova and Verilog source files)
- `%VAST5_HOME%/Config/vhdlkeywords.txt` (for VHDL source files)
- `%VAST5_HOME%/Config/ckeywords.txt` (for C source files)
- `%VAST5_HOME%/Config/vastkeywords.txt` (VaST functions and macros)

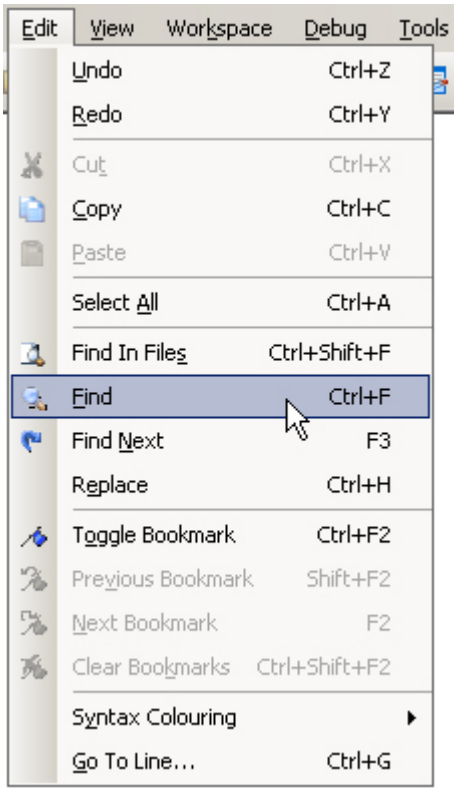
NOTE: **`%VAST5_HOME%`** stands for the directory where CoMET is installed.

To edit a list of keywords:

- Use the CoMET editor (or your favorite editor) to open the keyword file
- Add (or remove) keywords from the list
- Save the file

Editor features

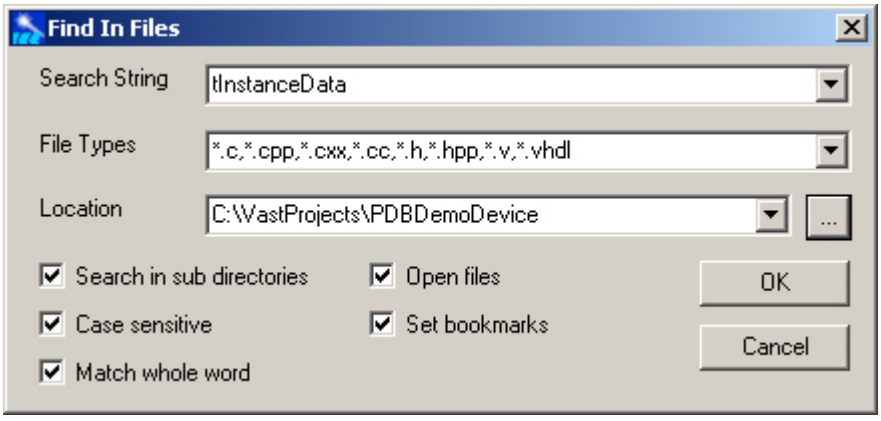
The Document window provides a number of useful features for editing code. With a source file open in the Document window, choose the Edit menu to see a list of available features.



Find in Files

This feature assists in finding function names or other specified strings in source code.

Choose **Edit/Find in Files** to locate specified text in files under a specified directory.



Find In Files dialog:

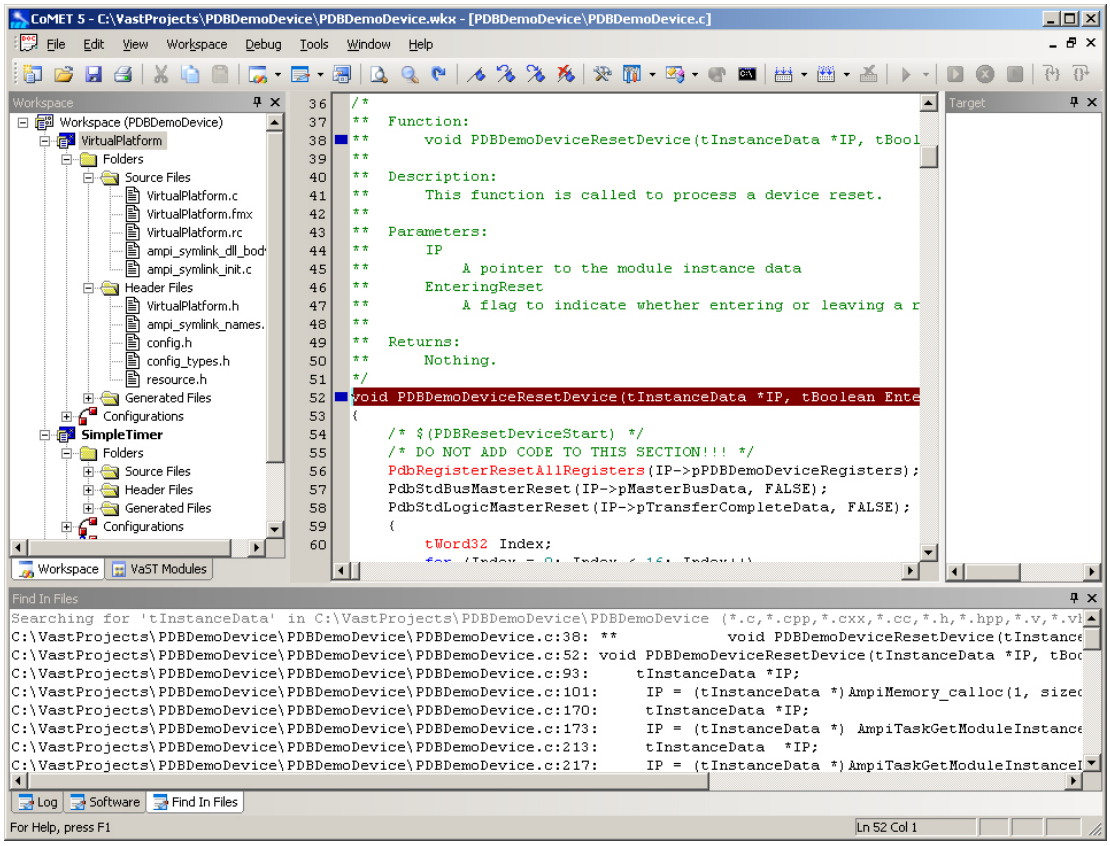
- **Search String**: Type the string for which you wish to search.
- **File Types**: Specify patterns for file types you wish to search for the string.
- **Location**: Specify the top level directory from which you wish to start searching.
- **Search in sub directories**: Check to search recursively through subdirectories of the Location directory.
- **Case sensitive**: Uncheck to ignore case in matching the search string to text in files
- **Match whole word**: Check to match only instances of the string with word delimiters on either side
- **Open files**: Check to open in the Document window all files in which a match is found
- **Set bookmarks**: Check to set bookmarks at all points where a match is found. This allows you to move from match to match by pressing the F2 key.
- Click **OK** to perform the search.

CoMET displays the results in the Find Output window.

- Double click a result line in the Find Output window to display the match in the file.

Note that matches in backup directories may be displayed, depending on the Find dialog **Location** and **Search in sub directories** property settings.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.


CoMET 5 - C:\VastProjects\PDBDemoDevice\PDBDemoDevice.wkx - [PDBDemoDevice\PDBDemoDevice.c]
File Edit View Workspace Debug Tools Window Help
Workspace
Workspace (PDBDemoDevice)
VirtualPlatform
Folders
Source Files
VirtualPlatform.c
VirtualPlatform.fmx
VirtualPlatform.rc
ampi_symlink_dll_bod
ampi_symlink_init.c
Header Files
VirtualPlatform.h
ampi_symlink_names.
config.h
config_types.h
resource.h
Generated Files
Configurations
SimpleTimer
Folders
Source Files
Header Files
Generated Files
Configurations
Workspace
VaST Modules
Target
/*
** Function:
** void PDBDemoDeviceResetDevice(tInstanceData *IP, tBool
** Description:
** This function is called to process a device reset.
** Parameters:
** IP
** A pointer to the module instance data
** EnteringReset
** A flag to indicate whether entering or leaving a r
** Returns:
** Nothing.
*/
void PDBDemoDeviceResetDevice(tInstanceData *IP, tBoolean Ente
{
/* $(PDBResetDeviceStart) */
/* DO NOT ADD CODE TO THIS SECTION!!! */
PdbRegisterResetAllRegisters(IP->pPDBDemoDeviceRegisters);
PdbStdBusMasterReset(IP->pMasterBusData, FALSE);
PdbStdLogicMasterReset(IP->pTransferCompleteData, FALSE);
{
tWord32 Index;
Find In Files
Searching for 'tInstanceData' in C:\VastProjects\PDBDemoDevice\PDBDemoDevice (*.c,*.cpp,*.cxx,*.cc,*.h,*.hpp,*.v,*.v
C:\VastProjects\PDBDemoDevice\PDBDemoDevice\PDBDemoDevice.c:38: ** void PDBDemoDeviceResetDevice(tInstance
C:\VastProjects\PDBDemoDevice\PDBDemoDevice\PDBDemoDevice.c:52: void PDBDemoDeviceResetDevice(tInstanceData *IP, tBo
C:\VastProjects\PDBDemoDevice\PDBDemoDevice\PDBDemoDevice.c:93: tInstanceData *IP;
C:\VastProjects\PDBDemoDevice\PDBDemoDevice\PDBDemoDevice.c:101: IP = (tInstanceData *)AmpiMemory_calloc(1, size
C:\VastProjects\PDBDemoDevice\PDBDemoDevice\PDBDemoDevice.c:170: tInstanceData *IP;
C:\VastProjects\PDBDemoDevice\PDBDemoDevice\PDBDemoDevice.c:173: IP = (tInstanceData *) AmpiTaskGetModuleInstance
C:\VastProjects\PDBDemoDevice\PDBDemoDevice\PDBDemoDevice.c:213: tInstanceData *IP;
C:\VastProjects\PDBDemoDevice\PDBDemoDevice\PDBDemoDevice.c:217: IP = (tInstanceData *)AmpiTaskGetModuleInstanceI
Log
Software
Find In Files
For Help, press F1
Ln 52 Col 1

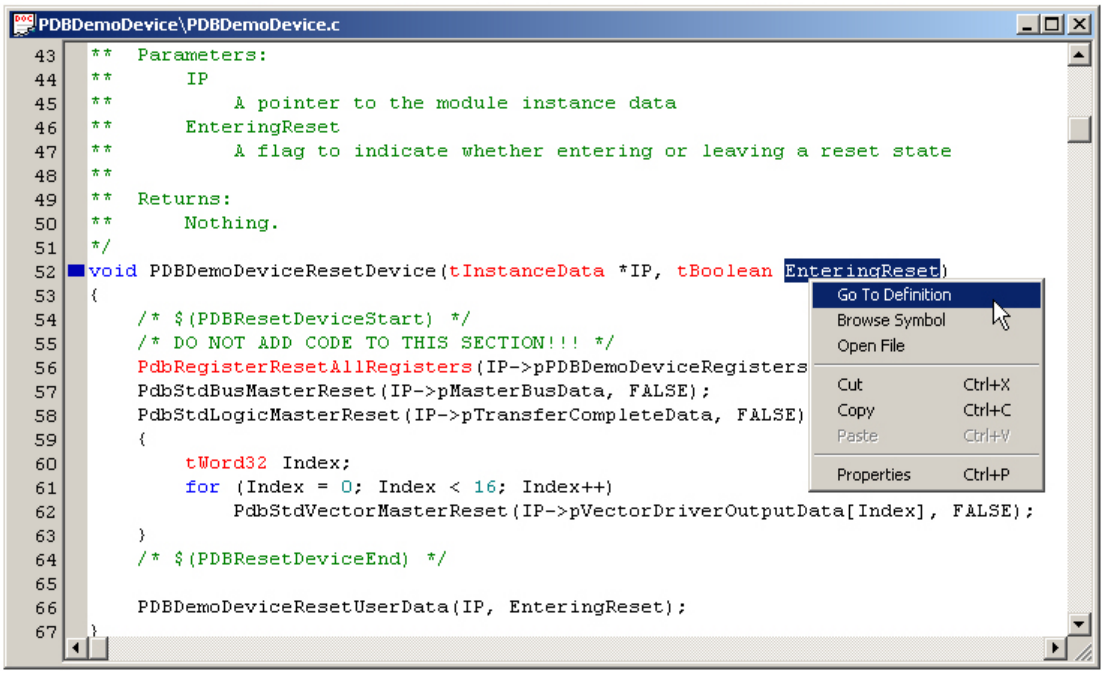
Locating Symbols with the Browse Database

CoMET provides browse features to assist in navigating code. The browse feature requires the browser database, which is built during a successful build of your model in Debug configuration. In the course of building the model, the output message appears:

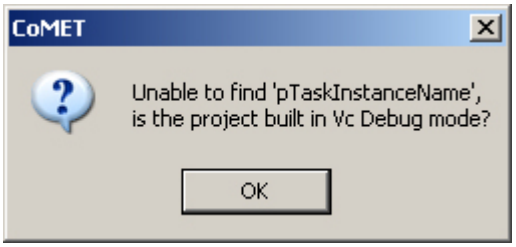
```
CoMET is building the browser database
```

Goto Definition

After the build completes, you can select an identifier in the code, right click and select **Goto Definition** from the context menu. CoMET opens the file containing the definition of the identifier.



NOTE: If you attempt to use the **Go To Definition** function without building the module a warning message will be displayed indicating that the identifier could not be found. An example of such a warning is shown below.

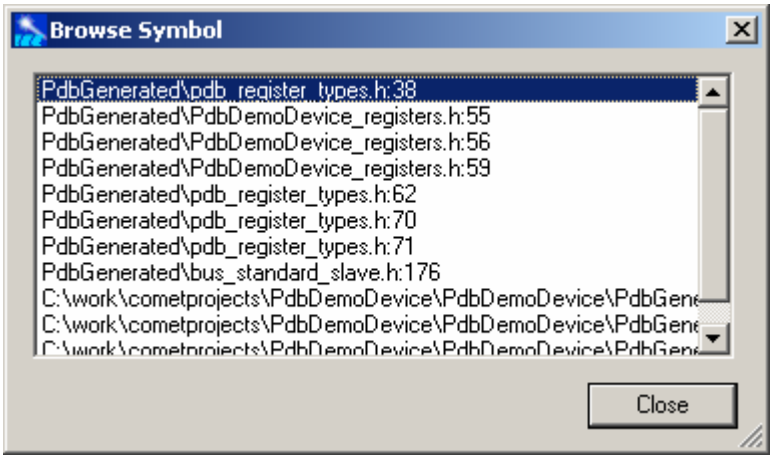


To fix this error simply build the module, see *Building the Workspace and Projects, page 137*.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

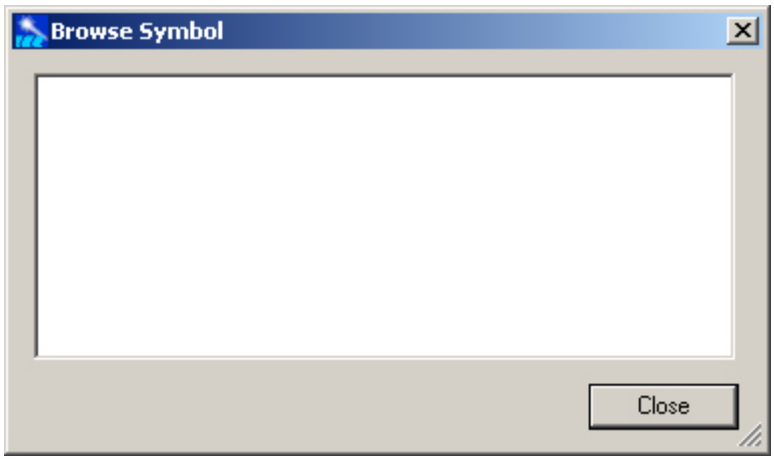
Browse Symbol

To see all occurrences of an identifier or symbol in code, select an identifier, right click and choose **Browse Symbol** from the context menu.



CoMET displays a list of occurrences of the symbol in code, showing file and line number. Click on an occurrence in the list to move to that file and line.

NOTE: If you attempt to use the **Browse Symbol** function without building the module an empty Browse Symbol dialog box will result, as shown below.



To fix this error simply build the module, see *Building the Workspace and Projects, page 137*.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Special Editing Keys

In the source code editor you can perform standard editing functions with commonly used hot keys. You can also toggle bookmarks and breakpoints, go to a specific line, and perform search and replace operations. Many of these options are available from the Edit menu as well as by hot key.

Key	Description
Navigation	
Ctrl-G	Go to line number
Ctrl-F2	Toggle bookmark on cursor line
F2	Move to next bookmark
Shift-F2	Move to previous bookmark
Ctrl-Shift-F2	Clear Bookmarks
Common editing functions	
Ctrl-Z	Undo
Ctrl-Y	Redo
Ctrl-X	Cut
Ctrl-C	Copy
Ctrl-V	Paste
Ctrl-P	Print
Find and replace	
Ctrl-F	Find
F3	Find next
Ctrl-H	Replace
Breakpoints	
F9	Toggle breakpoint on cursor line
Ctrl-Shift-F9	Remove all breakpoints

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Building the Workspace and Projects

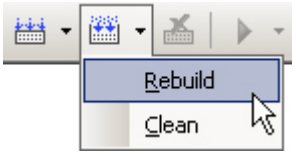
When you have defined the list of project source files, you can build the Virtual System Prototype. The build process compiles and links the source files and libraries, creating a simulator executable file (image).

To build the Virtual System Prototype:

- From the **Workspace** menu, select a **Build** (or **Rebuild**) option

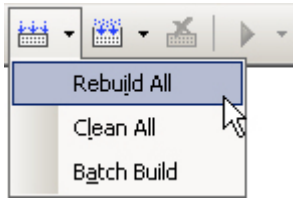
or

- Select the **Rebuild** tool bar button



or

- Select the **Rebuild All** tool bar button



The following describes the purpose of each Build option:

`Rebuild`	Builds the active configuration for the active project
`Clean`	Removes all built and CoMET generated files associated with the active project.
`Rebuild All`	Builds the active configuration for all projects in the workspace
`Clean All`	Removes all built and CoMET generated files for all projects in the workspace
`Batch Build`	Allows you to select which configurations and projects are to be built

CoMET begins building the system model. While CoMET is building the system model, the **Build** window displays messages generated by the compilers. To view a source file where a compile error has occurred, double-click the compiler message in the Build window.

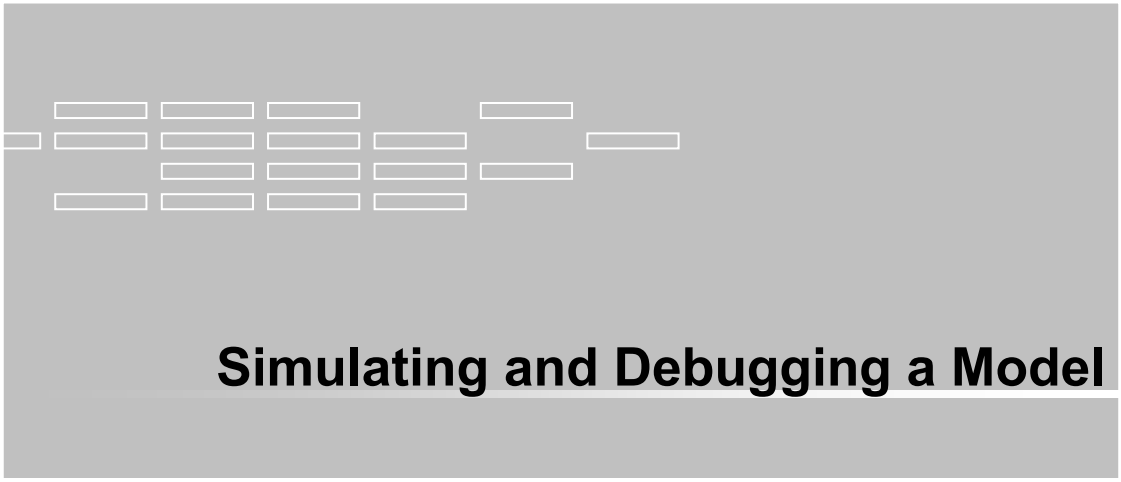
To stop the compilation before it terminates:

- From the **Workspace** menu, select **Stop Building**

or

- Select the **Stop** tool bar button

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.



Simulating and Debugging a Model

With CoMET, you can analyze the performance of your system model. The CoMET debugging tools are also useful for locating the source of bugs, analyzing how execution flows from one part of the system model to another, and observing how variables change as statements are executed.

CoMET allows you to execute the Virtual System Prototype in two modes: simulation and debug. A third execution mode, validation, is useful to certain customers for validating VPM behavior.

Choosing a Debugger

You can debug either Hardware Models, that is, VaST models, or Target code running on each VPM module instance in your platform. You determine what kind of debugging is performed at the start of each debug session. See *Executing in Simulation Mode and Debug Mode, page 141*.

Hardware Model Debugging

To debug your hardware model, you can use the CoMET internal debugger or an external debugger.

To choose an external debugger for Hardware Models, choose **Tools** and check the **External Debugger** menu item. The external debugger can be only VC6, VC7 or VC8.

NOTE:	You cannot use VC8 Express as an external debugger because VC8 Express does not support extensibility. For further information see the Microsoft Visual C++ Express website: http://msdn.microsoft.com/vstudio/express/visualC/default.aspx.

To choose an internal debugger for Hardware Models, choose **Tools** and ensure that the **External Debugger** menu item is unchecked.

The internal debugger used for Hardware Models depends on the active Configuration in the Project Workspace Configurations.

GDB Internal Debugger

If the selected Configuration requires the gcc compiler, e.g. Nova2GccDll, the internal debugger is GDB. The debugger output and command line interface appears in the software window, with a gdb prompt.

CDB Internal Debugger

If the selected Configuration requires the Vc compiler, e.g Nova2VcDll, the internal debugger is Cdb. The debugger output and command line interface appears in the software window.

Target Code Debugging

The external debugger used for Target code depends on the type of the VPM module instance on which the Target code runs. It also depends on your selection from among the supported cross-compilers for your Target code.

At the start of a debug session, for each VPM in your VSP you can independently select Debug or Simulation mode. See *Executing in Simulation Mode and Debug Mode*, below.

To interrupt execution during a Target Code debugging session, you use the Target Code break button for the VPM you wish to debug. See *Using the Target Debugger Break Button, page 144*.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

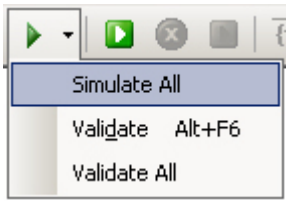
Executing in Simulation Mode and Debug Mode

To execute the Virtual System Prototype in simulation mode:

- Active the top model in the workspace
- From the **Workspace** menu, select **Simulate**

or

- Select the **Simulate All** from the **Simulate** tool bar button



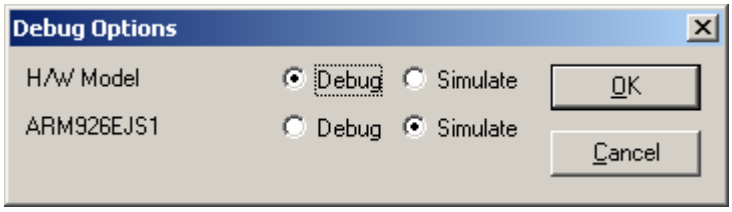
To execute the Virtual System Prototype in debug mode:

- Active the top model in the workspace
- From the **Debug** menu, select **Start Debug**

or

- Select the **Debug** tool bar button

CoMET may display dialogs requiring confirmation of build options. CoMET then displays the Debug Options dialog. The Debug options dialog allows you to select whether to perform Hardware model debugging or simulation, and whether to perform Target Code debugging or simulation on each of your VPMs.



- **H/W Model**: Select `Debug` to debug VaST models. Select `Simulate` to simulate the VaST models, i.e. run the simulation without debugging the VaST models in your platform.
- **ARM926EJS** (or other VPM instance): A set of Debug/Simulate options appears for each VPM module instance in the platform. Select `Debug` to debug Target code on the associated VPM. Select `Simulate` to run the Target code on the associated VPM without debugging.
- Click **OK**

The model begins executing and continues until it reaches an interruption to the execution, such as a breakpoint. See *Interrupting the Execution Flow,* below.

CoMET initiates execution of the Virtual System Prototype in a new process.

The Software window displays the output from the target executable running on the VPMs.

To stop debugging or simulation:

- From the Debug menu, select Stop Debug/Simulate

or

- Select the **Stop Debug** tool bar button

Interrupting the Execution Flow

You can interrupt execution in both Hardware Model debugging and Target Code debugging.

Setting Breakpoints in Hardware Model Debugging

You can interrupt the execution of your hardware models by setting breakpoints.

To set a breakpoint in the hardware or behavioral source code:

- In the C source file, click on the line of code.
- Click on the **Toggle breakpoint** tool bar button or press the **F9** key

A red diamond appears in the left window margin to indicate the breakpoint.

During debugging, the execution halts at the statement containing the breakpoint – before it is executed.

To clear a breakpoint:

- In the C source file, click on the line of code that has the breakpoint you wish to clear
- From the **Debug** menu, select **Stop At**

or

- Click on the **Toggle breakpoint** tool bar button

or

- Press the **F9** key

To clear all the breakpoints:

- From the Debug menu, select Remove All Breakpoints

or

- Click on the **Remove All Breakpoints** tool bar button

or

- Press the **CTRL+SHFT+F9** keys

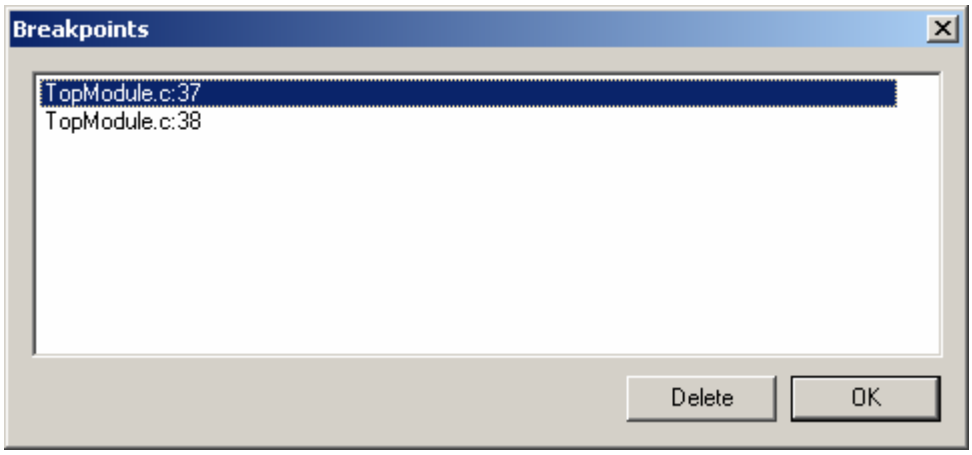
VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly
authorized in a written agreement with VaST.

Managing Breakpoints

You can add breakpoints at any statement within the source code of a module.

To see a concise list of breakpoints in the Breakpoints dialog box:

- From the **Debug** menu, select **Breakpoints**



Breakpoints Dialog

To jump to the location of a breakpoint:

- From the **Breakpoints** list box, select the breakpoint
- Click on the **OK** button.

You can also instruct the debugger to ignore all breakpoints without removing them from the breakpoint list (you can enable or disable all breakpoints as required)

Continuing Execution

Once execution has halted at a breakpoint, you can select one of three step modes to continue execution:

- Continue Debugging
- Step Into
- Step Over
- Step Out

Continue Debugging

The execution continues until the next breakpoint or the end of the code is reached.

To continue execution:

- From the **Debug** menu, select **Continue**

or

- Click on the **Continue** tool bar button

Step Into

The execution continues with the next statement in the code. If the statement includes a call to another user function in the project, the execution continues at (or steps into) the first line of the new function. In this mode, you can examine your entire code one statement at a time.

To step into a function:

- From the **Debug** menu, select **Step Into**

or

- Click on the **Step Into** tool bar button

Step Over

The execution continues with the next statement in the code. If the statement includes a call to another user function in the project, all of the code in the function is executed and the debugger continues execution at the following statement, unless the statement includes a VaST API function call or a breakpoint has been set elsewhere. In this case, execution halts at the breakpoint.

To step over a statement:

- From the **Debug** menu, select **Step Over**

or

- Click on the **Step Over** tool bar button

Step Out

The execution continues to the end of the current user function and halts at the line of code following the line of code that called the current function. If the current function includes another breakpoint, execution halts at the breakpoint.

To step over a statement:

- From the **Debug** menu, select **Step Out**

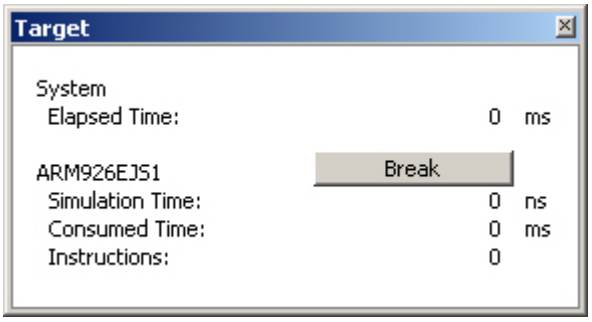
or

- Click on the **Step Out** tool bar button

Using the Target Debugger Break Button

During a Target debug session, a Break button appears for each VPM module instance selected for Target debugging. The Break button appears in the Workspace window Target tab. The appearance is similar to the button shown below.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly
authorized in a written agreement with VaST.



Click the Break button to break execution and launch the external debugger associated with the Target code. See the external debugger documentation for further information.

Validating VPM behavior

Validation mode is a special execution mode provided for customers who need to validate the Virtual Processor Model behavior against the microprocessor manufacturer test suite. Validation mode runs test target code on the Virtual Processor Model and returns a pass or fail status, indicating whether the VPM correctly conforms to the actual microprocessor behavior. Target code is a subset of the microprocessor manufacturer test suite.

NOTE: This option requires a specialized VSP and test suite code. For more details contact VaST Systems Technology Corporation.

Validate

To execute in Validation mode:

- In the Workspace menu, choose Validate

CoMET runs the target code and returns a Pass/Fail status in the output window.

Validate All

Validate All runs a set of microprocessor test routines. The set is specified by a list file.

To execute a set of Validation tests:

- In the Workspace menu, choose Validate All

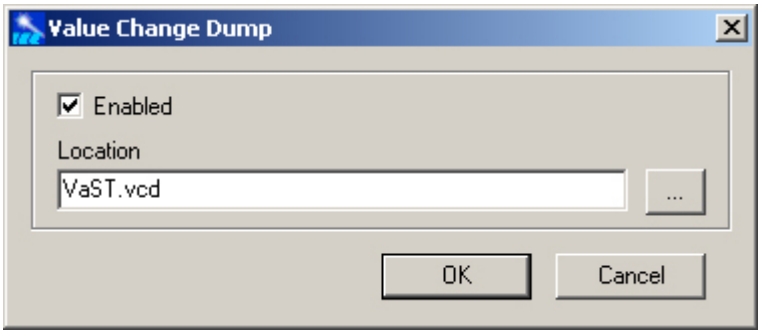
CoMET runs each of the specified target tests and returns a Pass/Fail status in the output window.

May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Obtaining a Value Change Dump (VCD)

To obtain a VCD:

- Choose **Tools/VCD Configuration**



- Check **Enabled** and specify the **Location** of the vcd file to be saved.

You can browse to the location. If you provide only the file name the file is saved in the current project directory.

The VCD is created during simulation. After simulation the VCD can be analyzed with standard VCD analysis tools.

Some modules, such as StdBus, provide parameters to enable or disable VCD output.

The size of the .vcd file is unlimited. You can indirectly control the size by enabling or disabling sources of VCD output.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly
authorized in a written agreement with VaST.

Building and Simulating in Batch Mode

CoMET allows you to build and simulate models in batch mode. After specifying workspaces and what actions are to be performed upon them in the Batch Mode dialog, you can run the batch from the Batch Mode dialog or from the command line. You can also run from the command line using a CoMET Batch file to specify options. See *CoMET Batch File, page 150*.

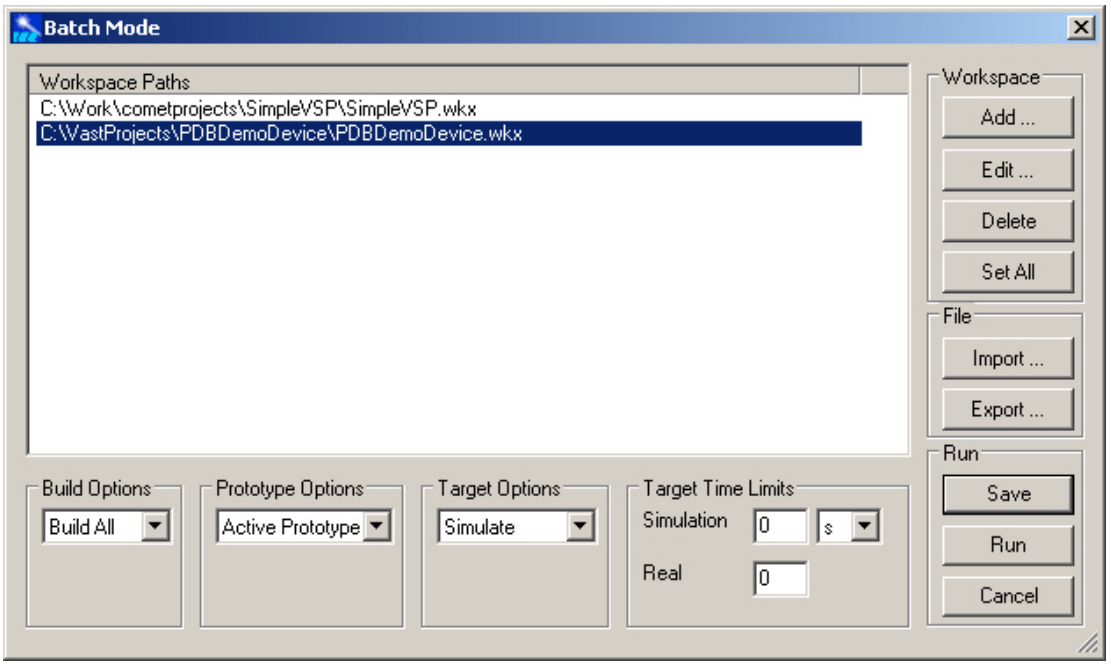
You can create only one set of batch mode options using the Batch Mode dialog. You can save a number of options sets as separate CoMET batch files.

Specifying Batch Mode Options using the Batch Mode Dialog

To specify Batch Mode options:

- Choose **Tools/Batch Mode**.

The Batch Mode dialog appears. It allows you to create a list of Workspaces by browsing for workspace (.wkx) files.



To add a workspace:

Click the Add button

- Browse for the workspace.

To delete a workspace:

- Select the workspace in the Batch Mode dialog Workspace Paths list
- Click Delete.

To Edit the workspace path as text:

- Select the workspace in the Batch Mode dialog Workspace Paths list
- Click Edit.

When the list is complete, you can select options to determine how to build and simulate all of the items in the list. For each workspace in the list there is a separate set of options.

Build options

- **None** - Build nothing.
- **Build All** - Build all projects in the workspace. This has the same effect as selecting the **Build All** option from menu or toolbar.
- **Rebuild All** - Rebuild all projects in the workspace. Same as selecting the **Rebuild All** option from menu or toolbar.

Prototype Options

- **Active Prototype** - Simulate only the active prototype
- **All Prototypes** - Simulate all prototypes in the workspace, one by one in the order in which they are listed in the workspace

Target options

- **None** - Simulate nothing
- **Simulate** - Simulate the active target in the prototype. This has the same effect as selecting the **Simulate** option from menu or toolbar.
- **Simulate All** - Simulate all targets in the workspace. This has the same effect as selecting the **Simulate All** option from menu or toolbar.
- **Validate** - Validate the active target in the prototype. This has the same effect as selecting the **Validate** option from menu or toolbar.
- **Validate All** – Validate all targets in the workspace. This has the same effect as selecting the **Validate All** option from menu or toolbar.
- **Debug** – Debug the active target in the prototype. This has the same effect as selecting the **Debug** option from menu or toolbar.
- **Debug All** - Debug all targets in the workspace. This has the same effect as selecting the **Debug All** option from menu or toolbar.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly
authorized in a written agreement with VaST.

Target Time Limits

You can set the maximum time for which the simulation runs.

- **Simulation:** Limit batch run to the specified number of simulation time units. Allowed units: [fs | ps | ns | us | ms | s] e.g. /SimTime=30ms : stop after 30 milliseconds of simulation time
- **RealTime**=*time* Limit batch run to the specified realtime seconds e.g. /RealTime=5 :stop after 5 seconds

Saving the batch

When the batch mode options are specified, you can save the batch mode specifications by clicking the **Save** button. The options are saved in the Windows Registry. CoMET can then be invoked from the command line. See *Command line batch mode* below.

Exporting a CoMET Batch File

You can export the Batch Mode dialog batch specifications as a CoMET 5 batch file. Click the **Export** button, browse to the required location and specify the file name to save. You can then reload the batch file using the **Import** feature, or specify the CoMET batch mode as an option when running CoMET in command line batch mode. See *Command line batch mode, page 150*.

Importing a CoMET batch file

If you have created a batch file in the CoMET 4 batch file or CoMET 5 batch file format, you can load it in the CoMET 5 batch dialog. Click the **Import** button and browse for the CoMET batch file. CoMET imports the options and displays it in the Batch dialog Workspace Paths list. You can then edit it using the editing features described above.

You can also run a CoMET 4 batch file from the command line. See *Command line batch mode using a CoMET 4 batch file, page 151*.

Running the batch

When the batch mode options are created, you can run the batch mode by clicking the **Run** button. Alternatively you can run the batch from the command line. See *Command line batch mode* below.

If anything causes a halt, such as errors in building or simulating, or manual intervention, batch mode ends.

You can use the **Stop Build** and **Stop Simulation** menu or toolbar items to stop execution.

Within each workspace, targets are built and simulated in the order in which they appear in the workspace projects tree.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Command line batch mode

CoMET can be started from a DOS command batch file with a /B parameter, specifying that CoMET is to execute with specified batch mode options. After a batch run invoked from the command line, CoMET closes.

You can specify batch mode options in the CoMET Tools/Batch Mode dialog. Alternatively you can specify them in a CoMET batch file.

When you run CoMET from the command line you have the following options:

```
CoMET [/B[atchMode] [<BatchFile.txt>] [/H[ide]] [/M[inimized]]
[/L[ogPath] <LogPath>]] | [<Workspace.wkx>] | [<File.txt>]
```

e.g.

```
c:\vast5_home\Bin\Comet.exe /b /h /l c:\logs
```

The above command line results in a CoMET run in batch mode with Batch Mode options specified in Tools/Batch Mode dialog and saved in the registry. The CoMET GUI does not appear. Log files of the Software, Log and Build output are saved in the c:\logs directory.

Batch Mode Options

B: batchmode - Run in batch mode. Use CoMET batch options, specified in CoMET Tools/BatchMode dialog, or optionally specify a CoMET batch file. See below

H: Hide - don't show the CoMET GUI

M: Minimized - minimize CoMET GUI

L: Logpath - specify a directory in which three logs are saved, corresponding to the CoMET Log, Build and Software output windows.

Non-Batch-Mode Options

Workspace.wkx : You may specify a Workspace on the command line. On launch CoMET opens the Workspace.

File.txt: You may specify a file other than a workspace on the command line. On launch CoMET opens the file in the Document window.

CoMET Batch File

A CoMET batch file is distinct from a DOS Batch file. It provides an alternative to specifying workspaces and options in the CoMET Tools/Batch Mode dialog.

It consists of lines of the form:

workspace.wkx /option1 /option2 ...

e.g.

```
C:\VastProjects\Fabric\base_sh2a_cif\base_sh2a.wkx /BuildAll
/AllPrototypes /SimulateAll /SimTime=20ms /RealTime=5
```


VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly
authorized in a written agreement with VaST.

Options can be specified with the minimum characters required for unique recognition, e.g. Simulate:Simulate SimTime:SimT

Options are:

/**Build**: Build the active project

/**BuildAll**: Build all projects

/**Rebuild**: Rebuild the active project

/**RebuildAll**: Rebuild all projects

/**Simulate**: Simulate the active project

/**SimulateAll**: Simulate all projects, one by one

/**ActivePrototype**: Run target for active prototype

/**AllPrototypes**: Run target for all prototypes one by one

/**SimTime**=*timeUnit* Limit batch run to the specified number of simulation time units. Allowed units: [fs | ps | ns | us | ms | s] e.g. /SimTime=30ms : stop after 30 milliseconds of simulation time

/**RealTime**=*time* Limit batch run to the specified realtime seconds e.g. /RealTime=5 :stop after 5 seconds

Command line batch mode using a CoMET 4 batch file

To use a CoMET 4 batch file, run CoMET in batch mode from the command line, specifying the name of the batch file:

```
comet /b filename
```

or

```
comet /batchmode filename
```

CoMET runs the batch file as described in *Command line batch mode* above.

For more details see *Application Note: Loading CoMET 4 Batch Files into CoMET 5.*

May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

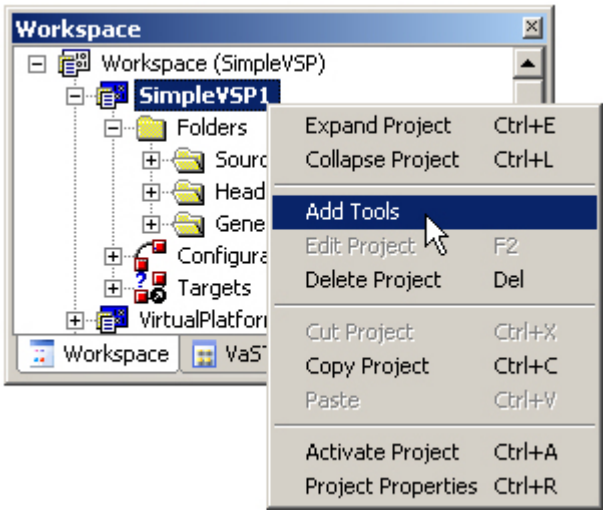
Running External Tools during Simulation

For each project, CoMET allows you to specify tools to be run immediately prior to each simulation. Such tools can perform functions such as communicating with the simulation process, to provide input and output.

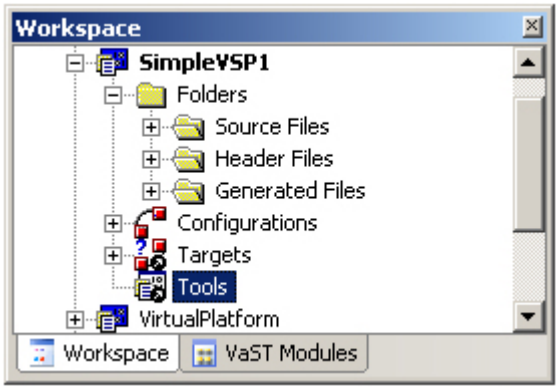
To specify a tool, first create a Tools node under the project node in the Workspace window.

To add a Tools node:

- Right click on the project icon in the Workspace window.
- Choose Add Tools from the context menu.



CoMET adds a Tools node.

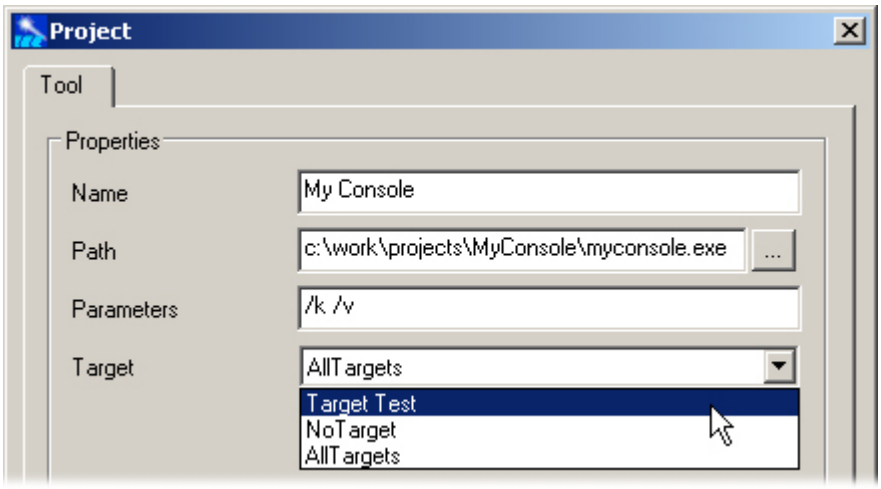


To add a tool to the Tools list:

- Right click on the Tools node.
- Select Add Tool from the context menu.

CoMET displays the **Project** dialog, **Tool** tab.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.



Project dialog, **Tool** tab, **Properties** panel:

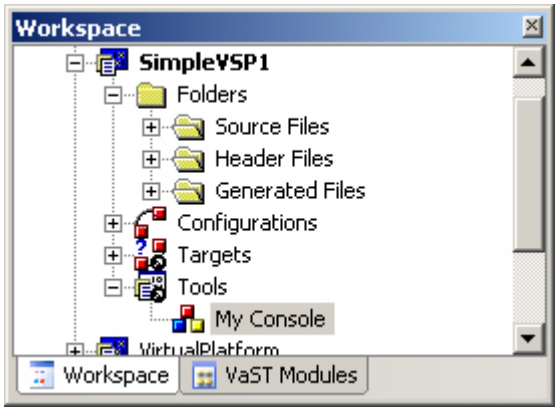
Name: Type the name under which you want the tool listed

Path: Browse for the path of the tool executable

Parameters: Specify command line parameters for the tool

Target: Choose the Targets with which the Tool is to be associated. You can choose a specific Target, no target, or all Targets.

Project dialog: Click **OK**.



CoMET displays the new Tool node under the project Tools node in the Workspace window. When you run a simulation, CoMET runs the tools in the Tools list first, and then runs the simulation.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Target Debugging Considerations

CoMET is designed to provide highly accurate timing during model simulation. Care must be taken in performing certain target debugging operations which may affect either simulation time or model state.

The modeler has a choice of whether to advance simulation time when the target debugger writes to or reads from a register. VaST recommends that the modeler does not advance simulation time when the target debugger reads from a register.

When the target debugger writes to a register, the choices are as follows:

If simulation time advances on the write, then debugging affects the accuracy of the simulation, and the model responds as expected to the write

If simulation time does not advance on the write, then debugging does not affect the accuracy of the simulation. However, if a change to the register value normally results in a model behavior response; this does not take place until simulation advances.

The choice depends on the modeler's requirements.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly
authorized in a written agreement with VaST.

Hot Keys

Many hot keys are documented in the CoMET menus next to the associated function, and in the tool tips that pop up when you hover over a tool bar button. This section gathers the information on hot keys and special mouse clicks under functional headings. Many keys operate according to the Windows conventions.

Hot Keys - All Windows

Key or Mouse	Context	Description
Shift-F7		View CoMET Configuration dialog
Ctrl-D		Go to command prompt
F1		Help

Hot Keys - All Windows - Files

Key or Mouse	Context	Description
Ctrl-N		New file
Ctrl-O		Open file
Ctrl-S		Save file
Ctrl-F4		Close top file in Document Window

Hot Keys - All Windows - Editing

Key or Mouse	Context	Description
Ctrl-Z		Undo
Ctrl-Y		Redo
Ctrl-X		Cut
Ctrl-C		Copy
Ctrl-V		Paste

Hot Keys - Viewing CoMET Windows

Key or Mouse	Context	Description
Ctrl-Alt-1		Toggle tool bar display
Ctrl-Alt-2		Toggle status bar display
Ctrl-Alt-3		Toggle Workspace Window display
Ctrl-Alt-4		Toggle Output Window display
Ctrl-Alt-5		Toggle Modules Window display

Hot Keys - Workspace Window

Key or Mouse	Context	Description
Enter	Item selected in workspace tree	Opens the selected item
Double Click	Double click on item in workspace tree	Opens the selected item
Ctrl-P	Item selected in workspace tree	Displays the properties of the selected item

Hot Keys - Document Window

Key or Mouse	Context	Description
Ctrl-Right-Click	Viewing .fmx or .pcx file, XML Table View, Ctrl-Right-Click on module instance	On selecting table view from context menu, restricts view to module instance
Ctrl-P	Editing source code	Print
Ctrl-F	Viewing source code	Find
F3	Viewing source code	Find again

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Ctrl-H	Viewing source code	Replace
F9	Viewing source code	Toggle breakpoint
Ctrl-Shift-F9	Viewing source code	Remove all breakpoints

Hot Keys - Document Window - Source Code Navigation and Bookmarks

Key or Mouse	Context	Description
Ctrl-G	Viewing source code	Go to line number
Ctrl-F2	Viewing source code	Toggle bookmark
F2	Viewing source code	Move to next bookmark
Shift-F2	Viewing source code	Move to previous bookmark
Ctrl-Shift-F2	Viewing source code	Clear Bookmarks

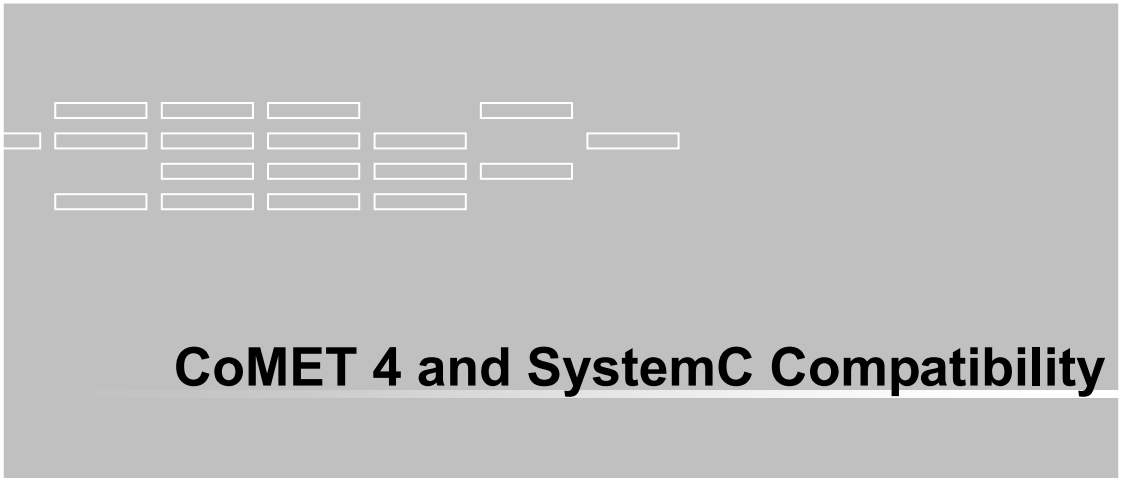
Hot Keys - Building Projects

Key or Mouse	Context	Description
Shift-F8		Build current project
Alt-F8		Build all projects
Ctrl-Del		Stop building

Hot Keys - Simulating and Debugging

Key or Mouse	Context	Description
Alt-F5		Simulate
Alt-F6		Validate
F5		Debug
F6		Continue (past breakpoint)
Shift-F5		Stop debugging or simulating
F11	Debugging, stepping through code	Step into code block
F10	Debugging, stepping through code	Step over code block
Shift-F11	Debugging, stepping through code	Step out of code block

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly
authorized in a written agreement with VaST.



CoMET 4 and SystemC Compatibility

Converting CoMET 4 projects

You can convert and edit VBus projects created in CoMET 4. A CoMET 5 VBus project can be built and simulated using similar procedures to those used in CoMET 4. For information on CoMET 5 support for VBus projects, see *Application Note: Converting CoMET 4 VBus Projects to CoMET 5 VBus Projects.*

Using the Matlab Generator

As in CoMET 4, you can generate Matlab source from a Matlab Interface (.mli) file. When CoMET 5 converts a CoMET 4 VBus project, it detects tools settings that specify the .mli file and includes the .mli file in the converted project.

To generate the Matlab source, choose **Tools/Matlab Generator**. This tool is only applicable when you are using a Workspace containing a converted CoMET 4 VBus project that included a Matlab interface.

After generating the source, according to the .mli file specifications, follow a similar procedure as that followed in CoMET 4 to externally compile the .dll. Link the .dll into the CoMET 5 VBus project and build as for other CoMET 5 projects.

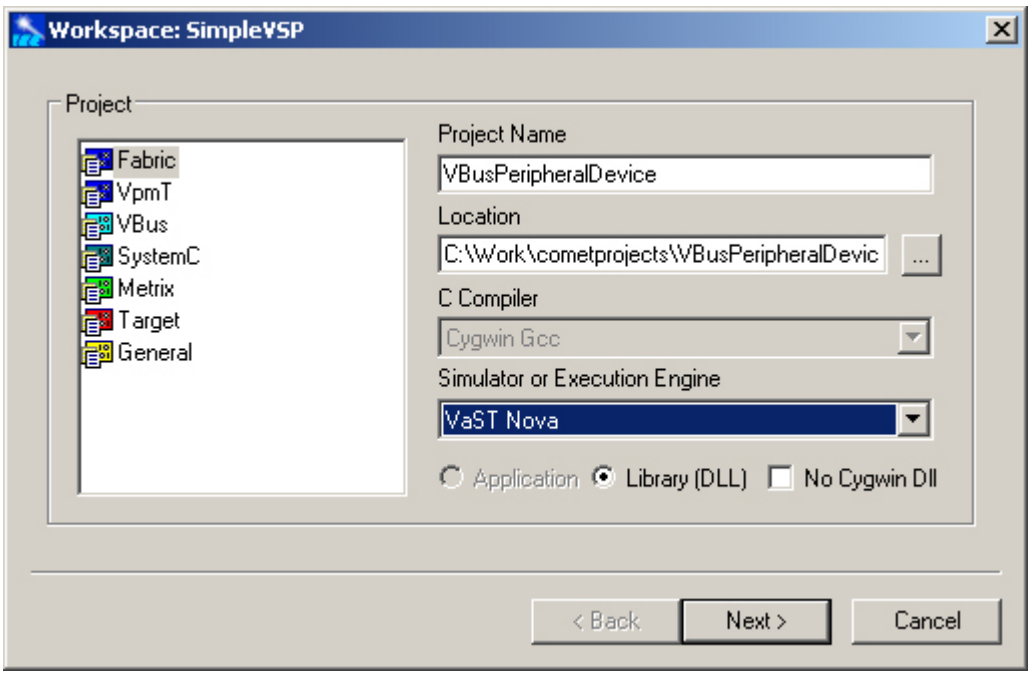
For more details see *Application Note: Converting CoMET 4 VBus Projects to CoMET 5 VBus Projects*.

Running a VBus Peripheral Model on CIF

You can import a VBus peripheral device model into CoMET 5. A VBus peripheral device model can then be instantiated in a CIF platform, and simulated as if it were a CIF model.

Creating a VBus Peripheral Model Project

Choose Workspace/Add New Project.



Project: Select the Fabric project type.

Project Name: Specify name matching the name of the existing VBus peripheral device module.

Location: Specify desired location for project files

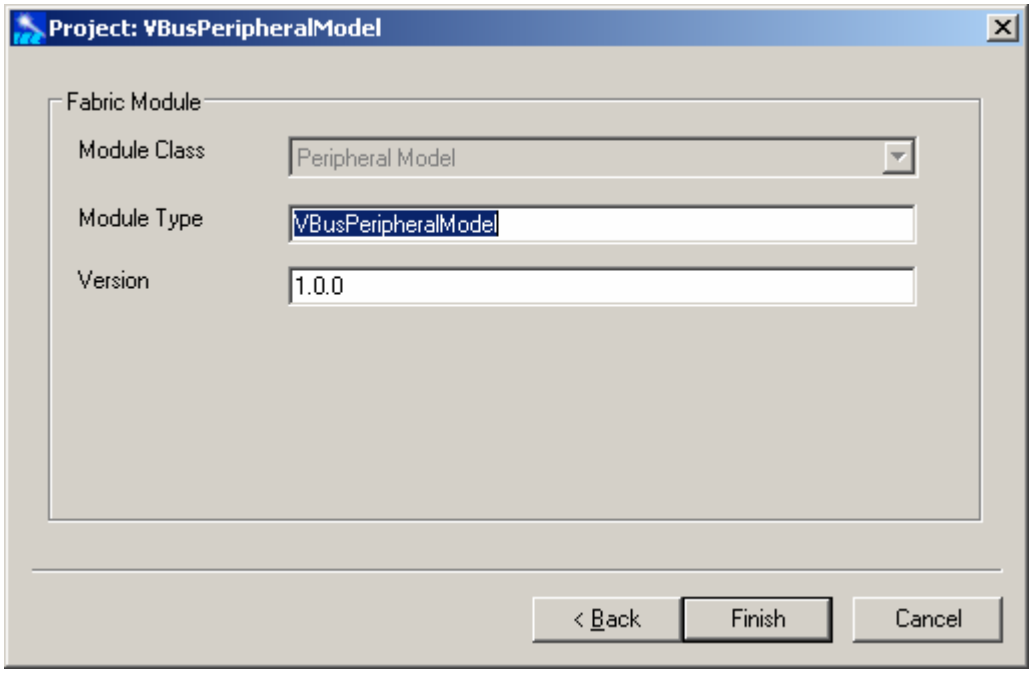
C Compiler: Select `Cygwin Gcc`

Simulator or Execution Engine: Select `VaST Nova`

No Cygwin DLL: Check this box to enable the project to use the No Cygwin Dll option.

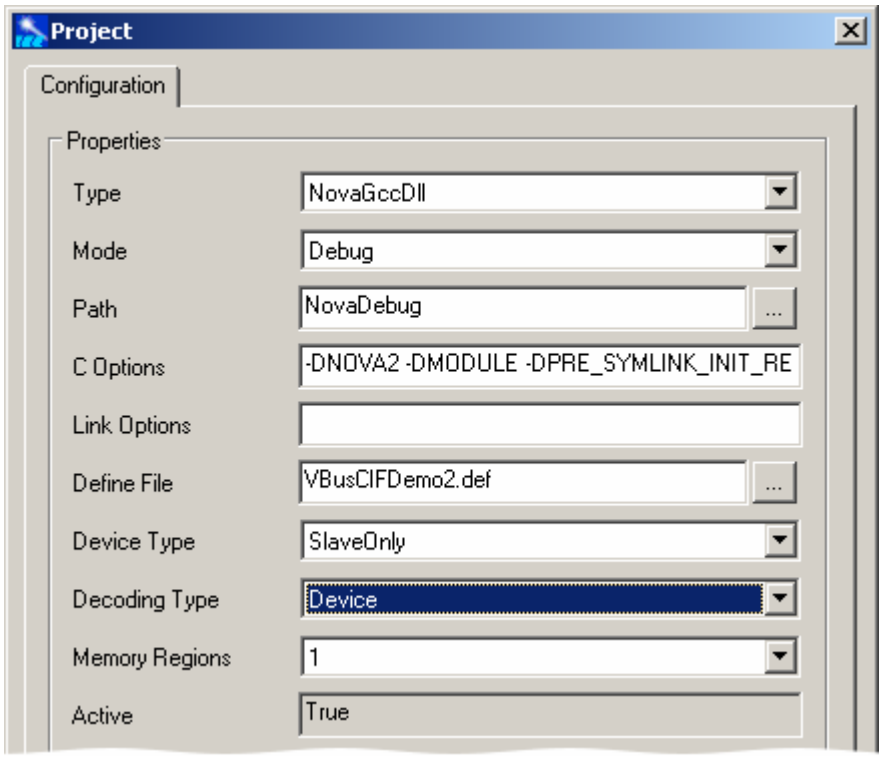
Click Next.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

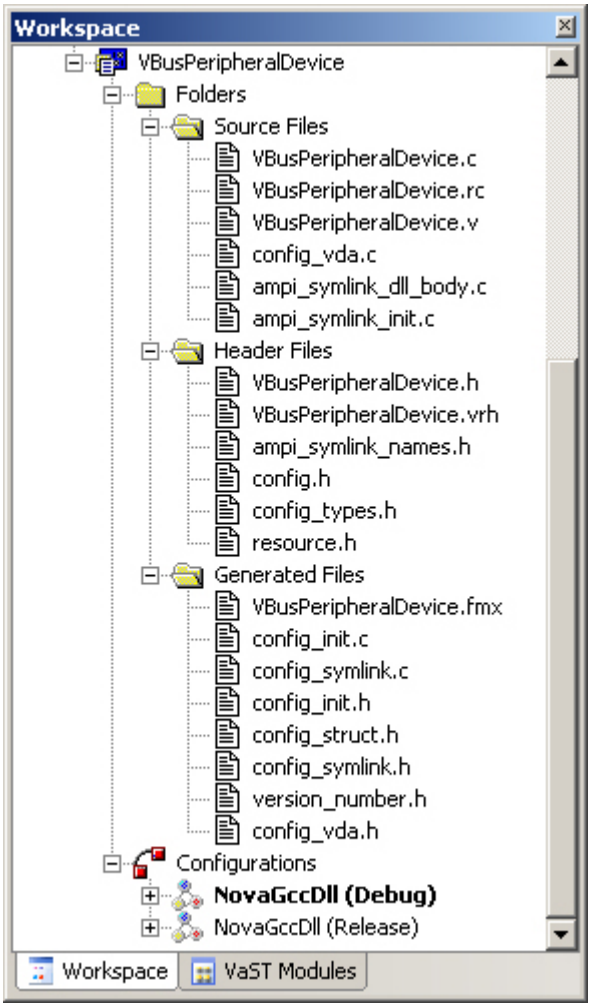


When the Simulator or Execution Engine is VaST Nova, the Fabric Module Class is restricted to Peripheral Model. Specify the Module type and version or accept the defaults. Click Finish.

Device Type, Decoding Type and number of Memory Regions are specified in the Build Configuration. The build type is NovaGccDll. See *Setting the Build Configuration, page 109*, and *VBus CIF Build Configuration Properties, page 113*.



CoMET creates a VBus Peripheral Model project from templates.



You can now replace the newly created files with the corresponding files from the existing VBus project:

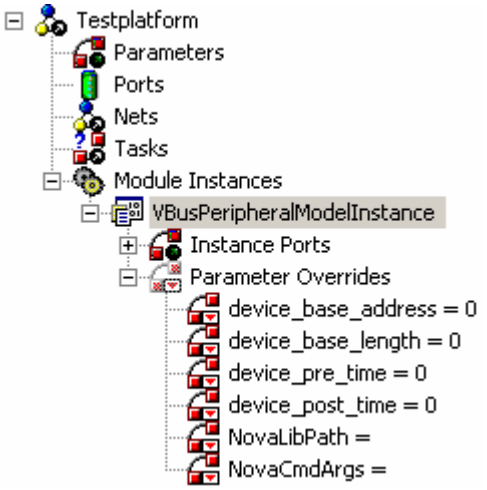
- `Source Files/`*`ProjectName`*`.c` – behavioral C for VBus device
- `Source Files/`*`ProjectName`*`.v` – HDL wrapper for VBus device
- `Header Files/`*`ProjectName`*`.h` – behavioral C header
- `Header Files/`*`ProjectName`*`.vrh` – HDL wrapper header

Provided you choose a project name to match the name of the existing VBus module, it is not necessary to alter the source files. You can edit them if you wish.

You can now build the VBus Peripheral Model.

When the VBus Peripheral Model is successfully compiled, a module instance of the model can be added to a CIF platform.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.



You can have only one VBus Peripheral Model in each VBus Peripheral Model project.

Parameters

CoMET automatically creates the following parameters for the VBus Peripheral Model. These appear in the module instance Parameter Overrides and the Prototype Configuration (.pcx) Parameter Overrides:

- `device_base_address`
- `device_base_length`
- `device_pre_time`
- `device_post_time`
- `NovaCmdArgs`

Specifying the VBus device type

The VBus device type is specified in the Project Configuration. Double click the VBus Device Project node in the workspace to open the Project Configuration dialog.

Simulating the VBus Peripheral Model

When you simulate a CIF VSP containing a VBus Peripheral Model, the VBus Peripheral Model is run by the Nova Execution Engine, and is co-simulating with other CIF Models in the VSP. You can specify the Nova Execution Engine command line parameters, with the NovaCmdArgs module instance parameter overrides.

Simulating a VBus Peripheral Model with FinSim

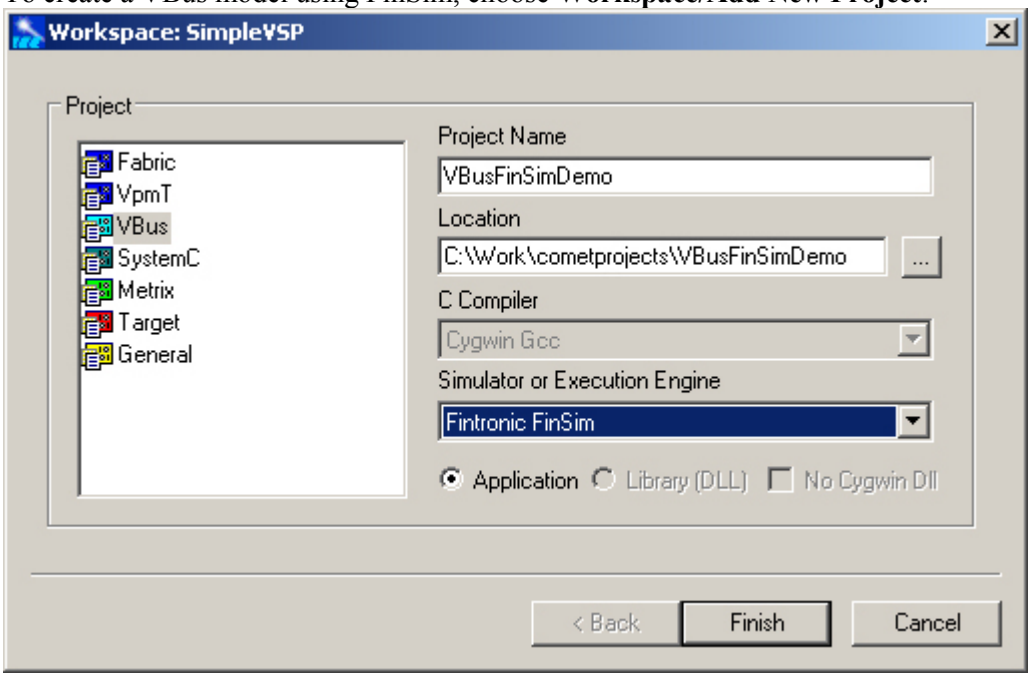
The FinSim verilog simulator requires a specific version of the GCC compiler, version 2.91.57.

May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

You cannot create a VBus module that is compatible with both CIF and FinSim, because to create a VBus CIF model, you must use GCC version 3.2, packaged with CoMET. This is incompatible with FinSim.

There is no Verilog debugger support.

To create a VBus model using FinSim, choose **Workspace/Add New Project**.



Project: Select the VBus project type.

Project Name: Specify the required name.

Location: Specify desired location for project files

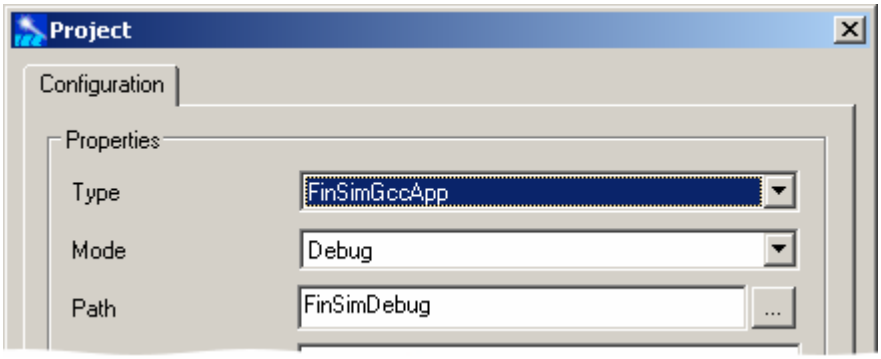
C Compiler: Select `Cygwin Gcc`.

In CoMET Configuration, Tool Locations tab, set Gcc to point to Gcc compiler version gcc_egcs_2.91.57. See *Tool Locations Configuration Settings, page 31*.

Simulator or Execution Engine: Select `Fintronic FinSim`

Click Next.

The Build Configuration for the VBus FinSim project shows a Build type of FinSimGccApp.



VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Creating SystemC Projects

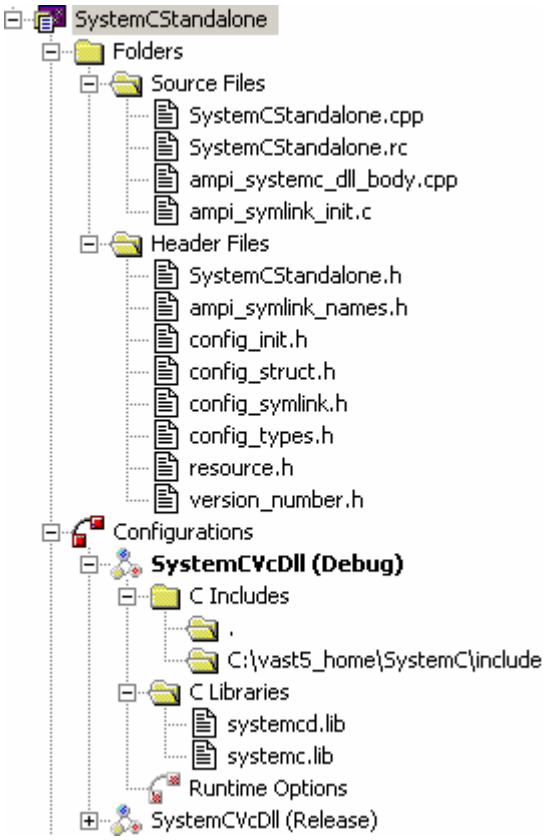
CoMET supports two kinds of SystemC Project. You can build a standalone SystemC project, using the SystemC project type. Alternatively you can build a project of type Fabric type, with a Module Class of Peripheral Model SystemC.

You need to choose SystemC libraries that correspond to the version of C++ compiler you are using. See *Choosing the SystemC Libraries for VC++6, VC++7, VC++8 and GCC, page 166*.

Creating a Standalone SystemC Project

Choose **Workspace/Add New Project**. In the Workspace dialog, Project panel, select a project type of `SystemC`. See *Adding a New Project, page 45*. Specify a name for the project and select other options as required.

CoMET creates a set of default files from templates.



You need to choose SystemC libraries that correspond to the version of C++ compiler you are using. See *Choosing the SystemC Libraries for VC++6, VC++7, VC++8 and GCC, page 166*.

Creating a CIF SystemC Project

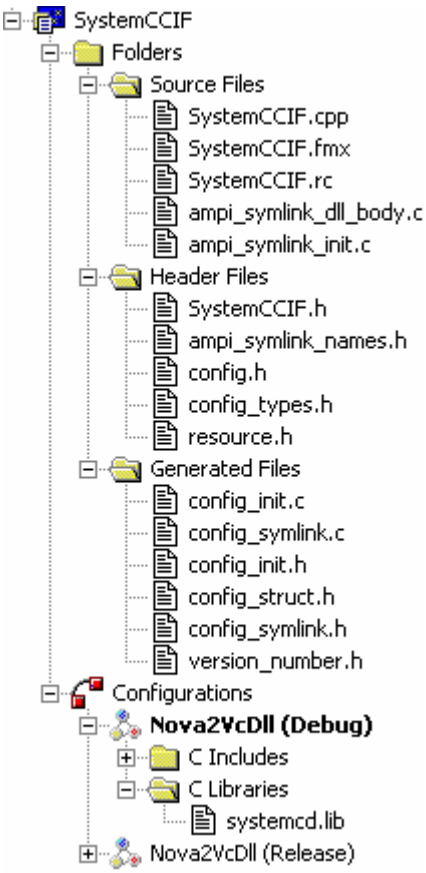
A module instance of a project of this type can be created in a CIF platform. To create a CIF SystemC project, choose **Workspace/Add New Project**. In the **Workspace** dialog, **Project**

May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

panel, select a project type of `Fabric`. See *Adding a New Project, page 45*. Specify a name for the project and select other options as required.

In the **Project** dialog, **Fabric Module** panel, **Module Class** field select `Peripheral Model SystemC`.

CoMET creates the project files from templates.



The template includes skeleton code for a SystemC adapter module with interfaces to both a Fabric bus and a sample SystemC device module.

To develop a module of this type, you need to be familiar with development in both Fabric and SystemC. For more details, see *Modeling Guide using the Communication and Infrastructure Fabric* and the application note *Interfacing a SystemC Device Model to a Fabric Platform*.

You need to choose SystemC libraries that correspond to the version of C++ compiler you are using. See *Choosing the SystemC Libraries for VC++6, VC++7, VC++8 and GCC, below*.

Choosing the SystemC Libraries for VC++6, VC++7, VC++8 and GCC

You need to choose SystemC libraries that correspond to the version of C++ compiler you are using. If you have chosen the Microsoft Visual C compiler, the template sets up the libraries for VC++ 6.

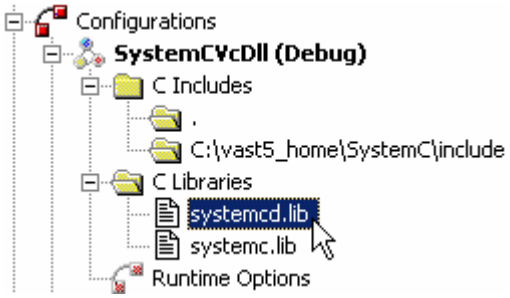
VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

If you are using VC++ 7, VC++ 8 or GCC (.net) you need to choose a different set of libraries. Libraries are found in the `%vast5_home%\SystemC\` directory. The libraries are as follows (the "d" in the library name denotes a Debug version):

Compiler	Library
Visual C++ v6	systemc.lib
Visual C++ v6	systemcd.lib
Visual C++ v7	systemc_vc7.lib
Visual C++ v7	systemcd_vc7.lib
Visual C++ v8	systemc_vc8.lib
Visual C++ v8	systemcd_vc8.lib
GCC	libsystemc.a

The libraries are specified within the Active Configuration. This is the highlighted configuration under the Configurations node in the Workspace Project tree. For example, in the screen picture below, the active configuration is **SystemCVcDll (Debug)**.

To change a library, double click the library file in the SystemC project Active Configuration **C Libraries** list. Browse and select the required library.

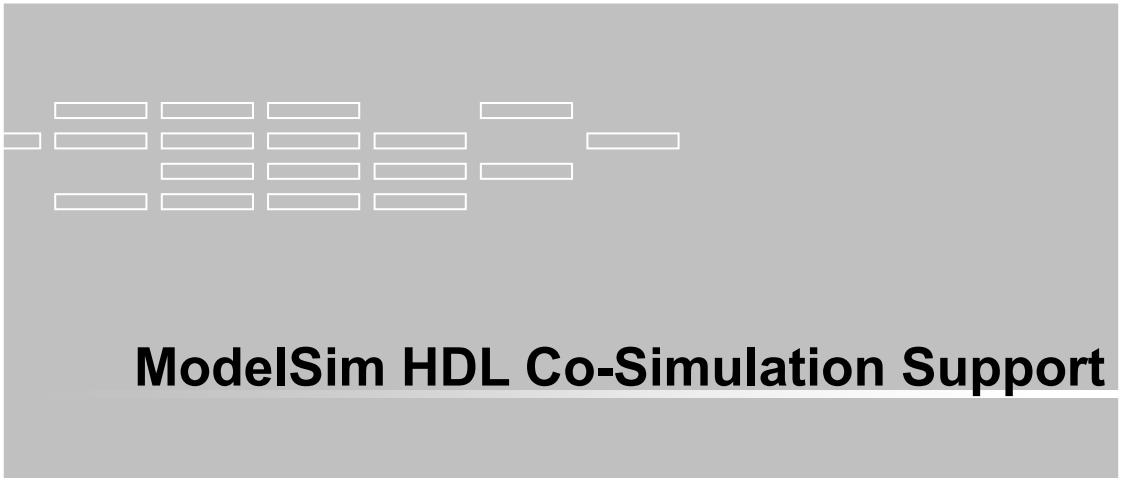


To add a library, right click the SystemC project Active Configuration **C Libraries** node and choose Add Library from the context menu. Browse and select the required library.

If there is no C Libraries node, you need to add it. Right click the Active Configuration node and choose **Add Libraries** from the context menu.

May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly
authorized in a written agreement with VaST.



ModelSim HDL Co-Simulation Support

Introduction to ModelSim HDL Co-Simulation Support

There are many RTL device models written in Verilog and VHDL and run on HDL simulators such as the Mentor Graphics ModelSim. The ModelSim HDL Co-Simulation support provides an easy way for existing RTL models written in Verilog or VHDL to run with CIF without the laborious task of rewriting and converting the models completely as CIF devices. This allows you to co-simulate your existing HDL device models with CIF with no or only minor modifications to the models.

VHDL Co-Simulation Support

Features and requirements

- ModelSim SE 6.0 or higher is required
- Pure VHDL models, as well as VHDL with Verilog instances are supported
- ModelSim Foreign Language Interface (FLI) with C models is supported
- A signal-level interface to the CIF AHB bus protocol is provided to interface VHDL AHB device models to the VaST AHB StdBus
- Debugging VHDL source is supported through the ModelSim graphic interface
- For VHDL array port types the element with the smallest index of the VHDL array port maps to index 0 of the corresponding CIF port, and the highest index of the VHDL array port maps to the highest index of the corresponding CIF port
- Only VHDL port types of `std_logic`, `std_ulogic`, `bit`, `std_logic_vector` (array of `std_logics`), `std_ulogic_vector` (array of `std_ulogics`), `bit_vector` (array of `bits`), `integer`, signed, unsigned and array of `integers` can be used to interface with CIF.

Mappings between VHDL and Fabric Port

The list of valid Fabric port types that may be mapped to the various VHDL port types is shown in the table below.

VDHL Port Type	Fabric Port Type	
	Default	**Other Valid User Selectable Types**
bit	StdLogic	StdVector32, StdClock
std_logic	StdLogic	StdVector32, StdClock
std_ulogic	StdLogic	StdVector32, StdClock
integer	StdVector32	StdClock
std_logic_vector	StdLogic array	StdVector32, StdClock
std_ulogic_vector	StdLogic array	StdVector32, StdClock
bit_vector	StdLogic array	StdVector32, StdClock
signed	StdLogic array	StdVector32, StdClock
unsigned	StdLogic array	StdVector32, StdClock
integer array	StdVector32 array	–

Notes:

- The mapping between VHDL std_logic type and Fabric StdLogic type is shown below

std_logic	StdLogic
'U'	StdLogicU
'X'	StdLogicX
'0'	StdLogic0
'1'	StdLogic1
'Z'	StdLogicZ
'W'	StdLogicW
'L'	StdLogicL
'H'	StdLogicH
'–'	StdLogicUnkown

- Overriding the default StdVector32 type and selecting StdLogic for integer VHDL ports or selecting StdClock for inout port is not allowed.
- If a StdClock is selected for a VHDL logic input port, VCST generates a clock signal for the VHDL logic port having the same clock period as the input StdClock.
- If a StdClock is selected for a VHDL logic output port, VCST will evaluate and output the clock period of the VHDL logic signal to the output StdClock port.

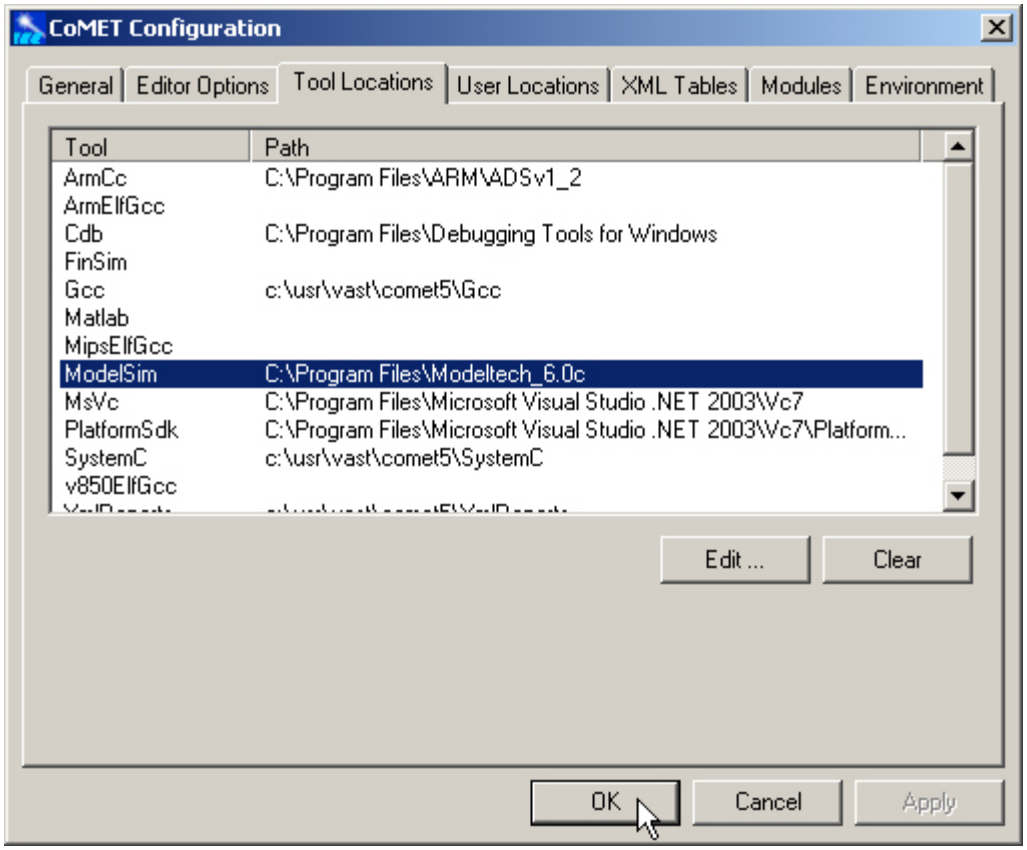
VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

- If StdClock is selected for a VHDL integer input port, the VHDL integer port will be provided with the clock period value of the StdClock.
- If StdClock is selected for a VHDL integer output port, the clock period value of the StdClock will be assigned with the value of the integer port outputted.
- If StdVector32 is selected for a VHDL output port of std_logic, std_ulogic or array of these types then a value of H (Weak High) or 1 (Driven High) will be mapped to a value of 1 in the StdVector32. All other values will be mapped to 0. This also applies to arrays of std_logic and std_ulogic. In this case the corresponding bit value will be set accordingly, as shown n the following example.
  A std_logic_vector value of "H0LZX1" will be mapped to the StdVector32 value of:
  0000 0000 0000 0000 0000 0000 0010 0001.
- If StdVector32 is selected for a VHDL input port of std_logic, std_ulogic or array of these types then a bit value of 1 in the StdVector32 will be mapped to a 1 (Driven High) and a bit value of 0 will be mapped to 0 (Driven Low).
- For details on how to select Fabric port types for mapping to VDHL ports see *Selecting a Fabric Port Type for a VDHL Port*, *page 176*.

Running a VHDL Model on CIF

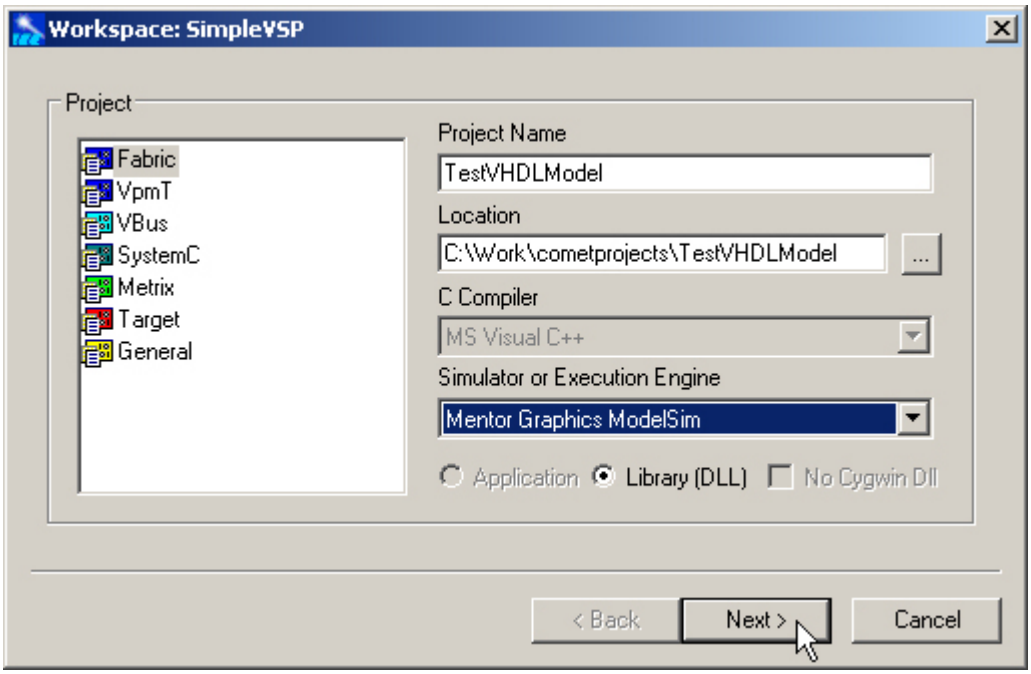
Tool Locations Configuration Settings for VHDL Model

In CoMET Configuration, Tool Locations tab, set ModelSim to point to ModelSim home directory. See *Tool Locations Configuration Settings, page 31*.



Creating a VHDL Model Project

Choose **Workspace/Add New Project**.



Project: Select the Fabric project type.

Project Name: Specify name of the VHDL Project.

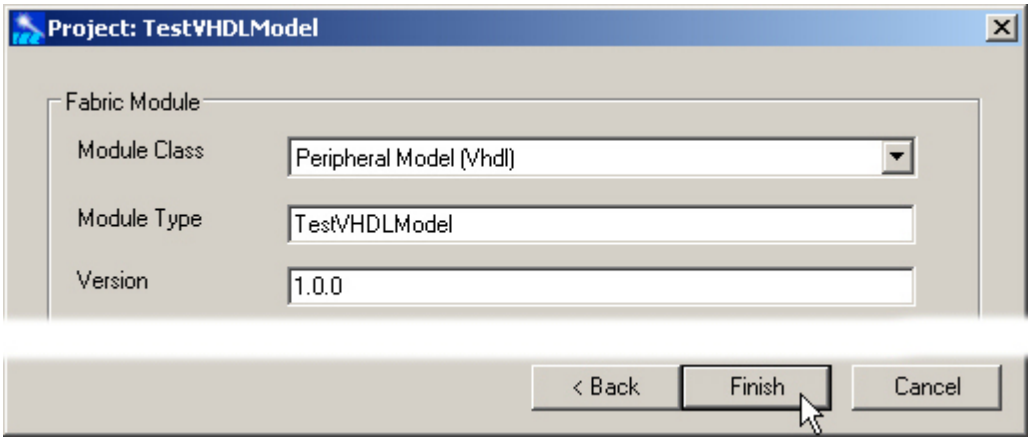
Location: Specify desired location for project files

C Compiler: Select `MS Visual C++`

Simulator or Execution Engine: Select `Mentor Graphics ModelSim`

Click Next.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly
authorized in a written agreement with VaST.



Module Class: Select `Peripheral Model (Vhdl)` for a VHDL project.

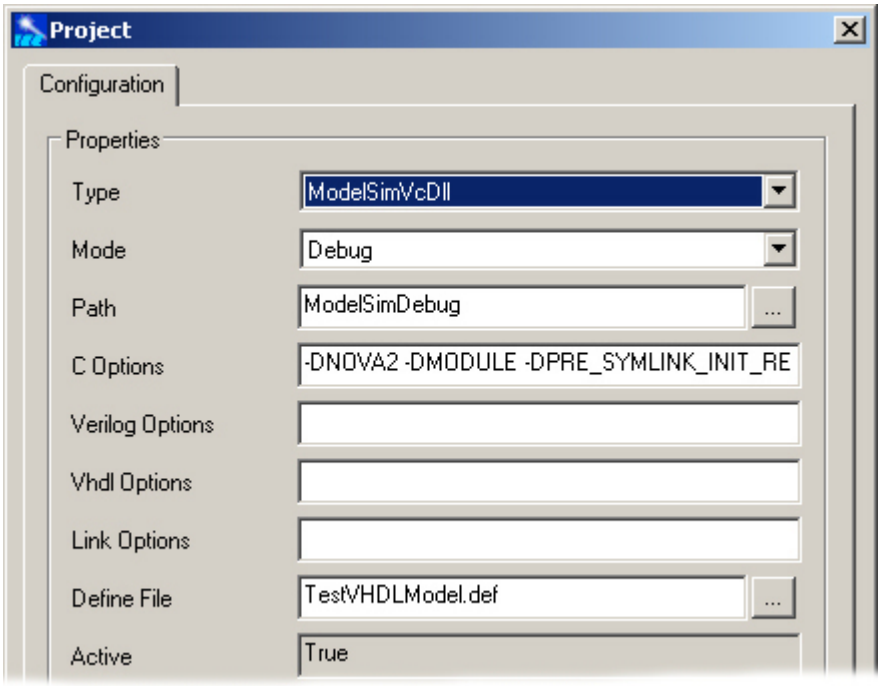
Module Type: Specify the name of the VHDL entity in the Module Type field or accept the default.

Version: Specify the version or accept the default.

Click Finish.

NOTE: The Module Type has to match the entity name of the VHDL entity. The top VHDL entity may instantiate other VHDL entities and/or Verilog modules in the project.

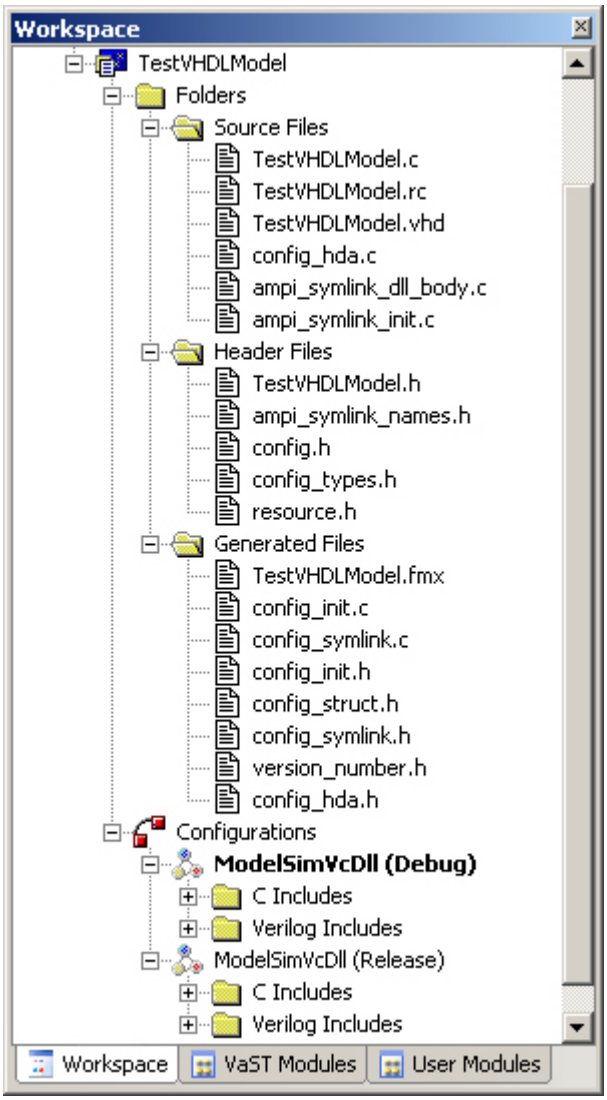
Double-click the Build Configuration to open the Configuration window:



Specify the Verilog complier options for Vlog in the Verilog Options field if there are Verilog modules in the project.

Specify the VHDL complier options for Vcom in the VHDL Options field.

CoMET creates a VHDL Model project from templates.



VHDL Model

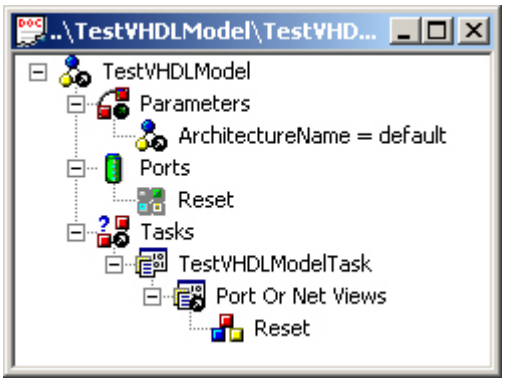
You can now replace the newly created files with the corresponding files from the existing VHDL project:

- Source Files/ProjectName.c – behavioral C for HDL device
- Source Files/ProjectName.vhd – VHDL source file
- Header Files/ProjectName.h – behavioral C header

Alter the above source files as required. You can also add other VHDL or Verilog source files to the project.

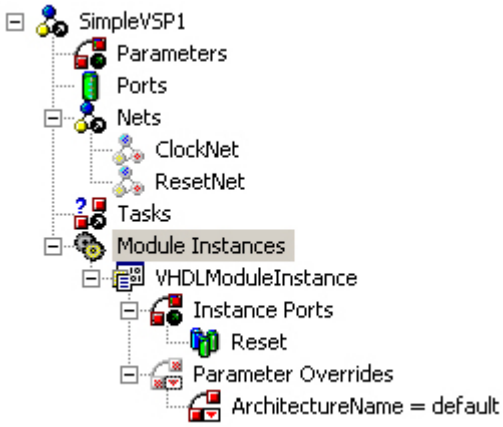
You can now build the VHDL Model.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.



Xml tree view of automatically generated VHDL .fmx file

When the VHDL Model is successfully complied, a module instance of the model can be added to a CIF platform.



Parameters

CoMET automatically creates the following parameters for the VHDL Model. These appear in the module instance Parameter Overrides and the Prototype configuration (.pcx) Parameter Overrides:

- `ArchitectureName` – Specify the architecture to be used in each instance of this entity. If the value of "default" is accepted then the simulator will choose the default architecture to be used.

For each generics declared in the VHDL model, a corresponding parameter of the same name will be created automatically.

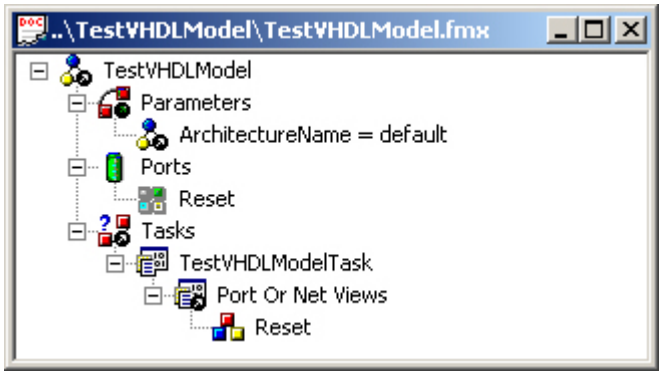
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Selecting a Fabric Port Type for a VDHL Port

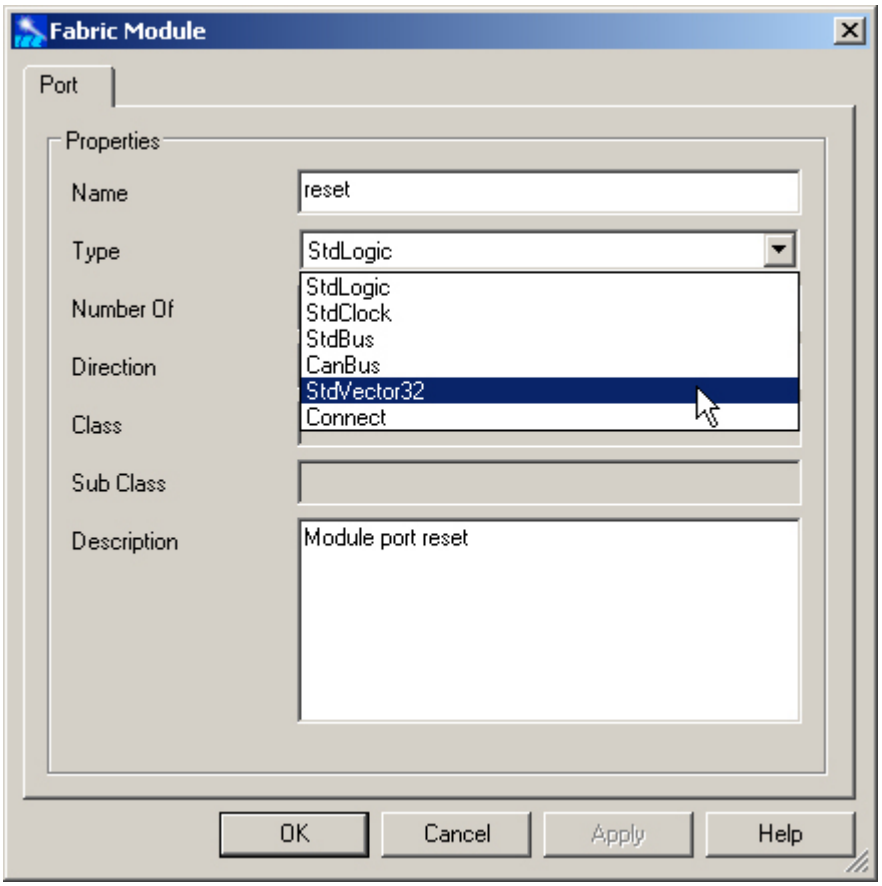
Note: This can only be done after the VHDL source files have been successfully complied and the .fmx file has been successfully updated with the list of ports in the VHDL entity.

To select a particular Fabric Port Type for a VHDL Port:

- Double-click on the **Generated Files/ProjectName.fmx** file. This will open the generated .fmx file document window.



- Double click on the port for which you want to set the Fabric Port Type.



VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

- Select the Fabric Port Type from the **Type** pull down menu. Refer to *Features and requirements, page 169*, for a list of valid Fabric Port types which can be selected for particular VHDL port types.
- Click OK to accept the selection.

Setting the Runtime Options

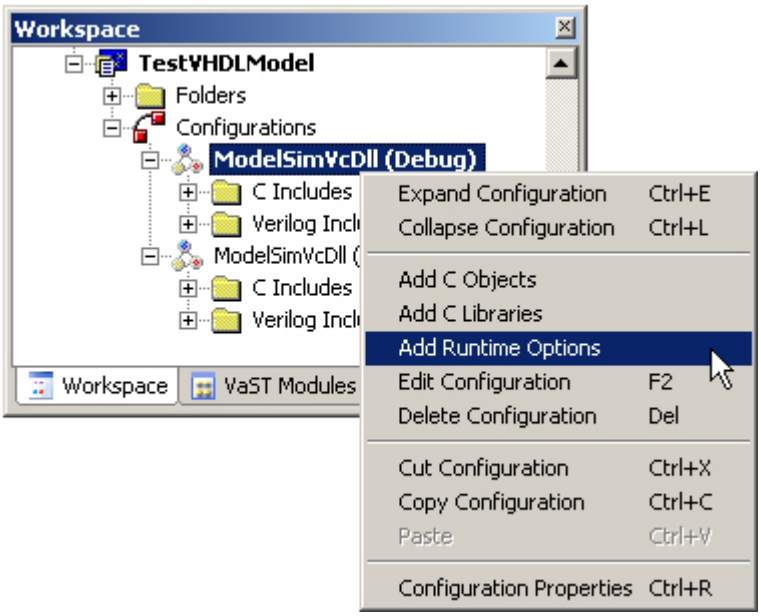
The build configuration of the top module supports a set of runtime options. Runtime Option properties vary for different build configuration types (See *Build Configuration Types, page 110*).

Adding Runtime Option Properties

To gain access to view and change Runtime Options you need to create a Runtime Options node in the Workspace window, under the Configuration node. CoMET creates a Runtime Options node automatically only for top level Fabric projects and VBus projects. You can manually add a Runtime Options node.

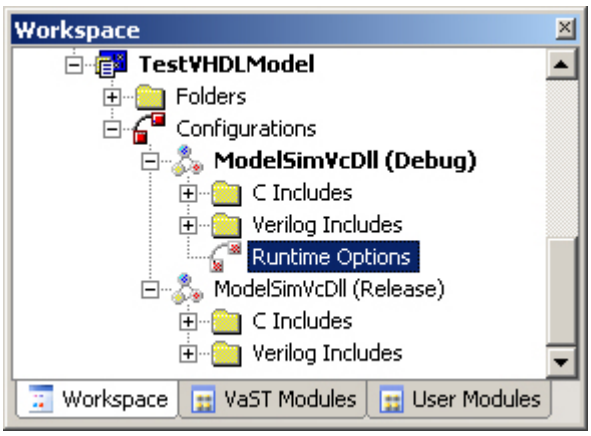
To add runtime options:

- Right click on the appropriate build Configuration node
- Choose **Add Runtime Options** from the context menu.



CoMET creates a new Runtime Options node under the build Configuration node.

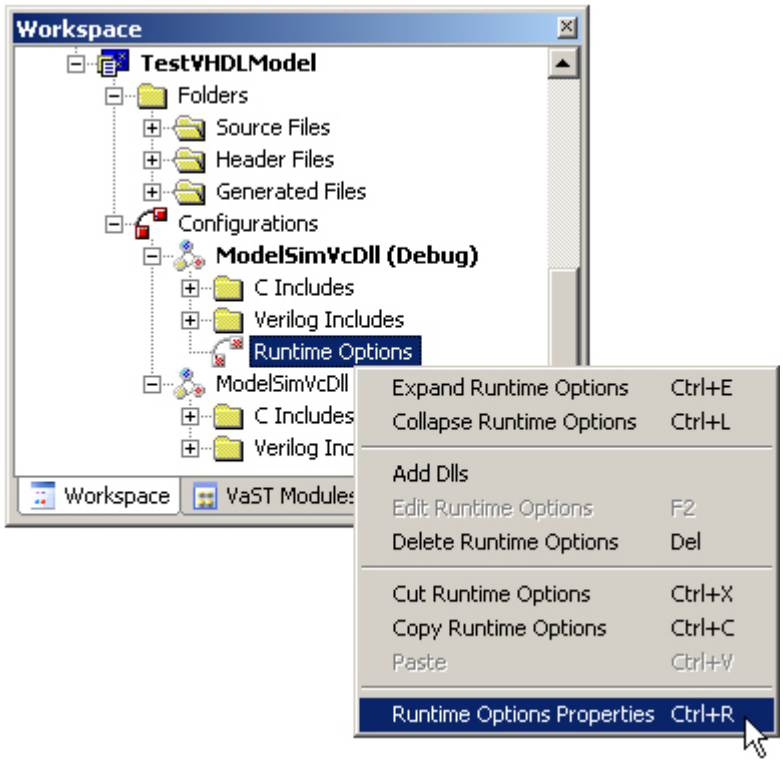
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.



Viewing and Setting Runtime Options Properties

To set the runtime options for the top module:

- Right mouse-click on Runtime Options
- Select **Runtime Options Properties**

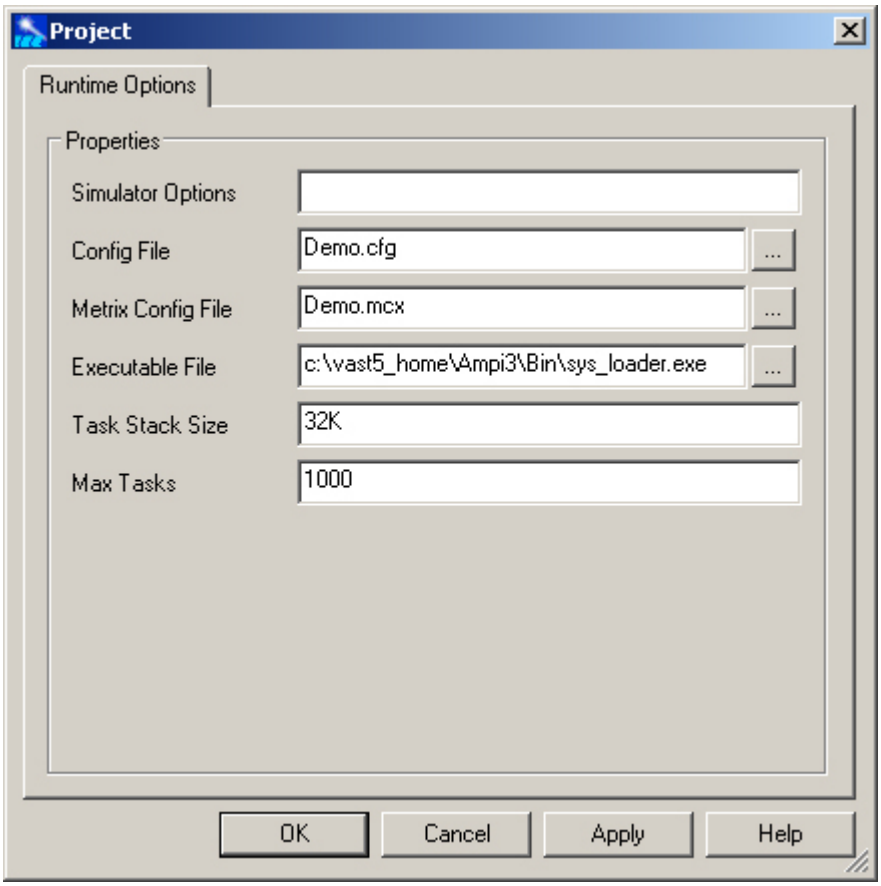


or

- Double-click on **Runtime Options**

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly
authorized in a written agreement with VaST.

VHDL Project Runtime Options



Simulator Options: These options are passed to the hardware simulator. For the ModelSim Co-Simulation Support there are some additional options you can specify.

1. `-script <commandfile>` : used to specify the name of the file containing the ModelSim simulation commands to be executed.
2. `-vcddump <vcdfilename>` : used to activate the VCD dump of all the signals in the HDL hierarchy to the specified file.
3. `-deltamode` : any HDL port signal change will only be propagated to the CIF at the next delta. If this option is not specified all the signal changes will only be propagated to CIF at the end of the delta cycles.
4. `-sharemem <size>` : Amount of share memory, in bytes, to be allocated for sharing information (for example Port states) between the Nova-2 scheduler and the ModelSim simulator. A default value of 512 K of share memory is allocated.

Config File: The Architecture Configuration File for this project.

Metrix Config File: The path of the associated Metrix Configuration (.mcx) file. See the *Metrix Configuration File User Guide.*

Executable File: This specifies the file that calls the Fabric .dll.

Task Stack Size: Default 32K. May be increased for large projects.

Max Tasks: Default 1000. May be increased for large projects.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Verilog Co-Simulation Support

Features and requirements

- ModelSim SE 6.0 or higher is required
- Pure Verilog models as well as Verilog with VHDL instances are supported
- ModelSim Procedural Language Interface (PLI) with C models is supported
- A signal-level interface to the CIF AHB bus protocol is provided to interface Verilog AHB device models to the VaST AHB StdBus
- Debugging Verilog source is supported through the ModelSim graphic interface
- Due to the limitation of the PLI implemented in the current version of ModelSim, Verilog bidirectional (inout) port types are not currently supported.
- For Verilog vector port types, the element with the smallest index maps to index 0 of the corresponding Fabric port, and the highest index of a Verilog vector port maps to the highest index of the corresponding Fabric port
- Array index that is a negative number or an expression evaluating to a negative number in a vector port declaration of a top Verilog module port is currently not supported. Only the +, –, *, / and % (mod) arithmetic operators are supported in expression used as array index of a vector port declaration.

Mappings between Verilog and Fabric Port

The list of valid Fabric port types that may be mapped to the various Verilog port types is shown in the table below.

Verilog Port Type	**Fabric Port Type**	
	Default	**Other Valid User Selectable Types**
Logic (scaler)	StdLogic	StdVector32, StdClock
Logic (vector)	StdLogic array	StdVector32, StdClock

Notes:

- Selecting StdClock for inout port is not allowed.
- If a StdClock is selected for a Verilog input port, a clock signal for the Verilog port having the same clock period as the input StdClock will be generated.
- If a StdClock is selected for a Verilog output port, the clock period of the Verilog signal will be evaluated continuously and outputted to the StdClock output port.
- If StdVector32 is selected for a Verilog output port, then the corresponding bits of the Verilog signal are mapped as follows:

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Verilog	StdVector32 Bit
HiZ	0
Sm0	0
Sm1	1
SmX	0
Me0	0
Me1	1
MeX	0
We0	0
We1	1
WeX	0
La0	0
La1	1
LaX	0
Pu0	0
Pu1	1
PuX	0
St0	0
St1	1
StX	0
Su0	0
Su1	1
SuX	0

- If StdLogic is selected for a Verilog output port, then the Verilog states are mapped to the StdLogic states as follows:

Verilog	StdLogic
HiZ	StdLogicZ
Sm0	StdLogicL
Sm1	StdLogicH
SmX	StdLogicW
Me0	StdLogicL
Me1	StdLogicH
MeX	StdLogicW
We0	StdLogicL
We1	StdLogicH
WeX	StdLogicW
La0	StdLogicL

May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Verilog	StdLogic
La1	StdLogicH
LaX	StdLogicW
Pu0	StdLogicL
Pu1	StdLogicH
PuX	StdLogicW
St0	StdLogic0
St1	StdLogic1
StX	StdLogicX
Su0	StdLogic0
Su1	StdLogic1
SuX	StdLogicX

- If StdLogic is selected for a Verilog input port then the StdLogic states are mapped to the Verilog states as follows:

StdLogic	Verilog
StdLogicU	StX
StdLogicX	StX
StdLogic0	St0
StdLogic1	St1
StdLogicZ	HiZ
StdLogicW	PuX
StdLogicL	Pu0
StdLogicH	Pu1
StdLogicUnkown	StX

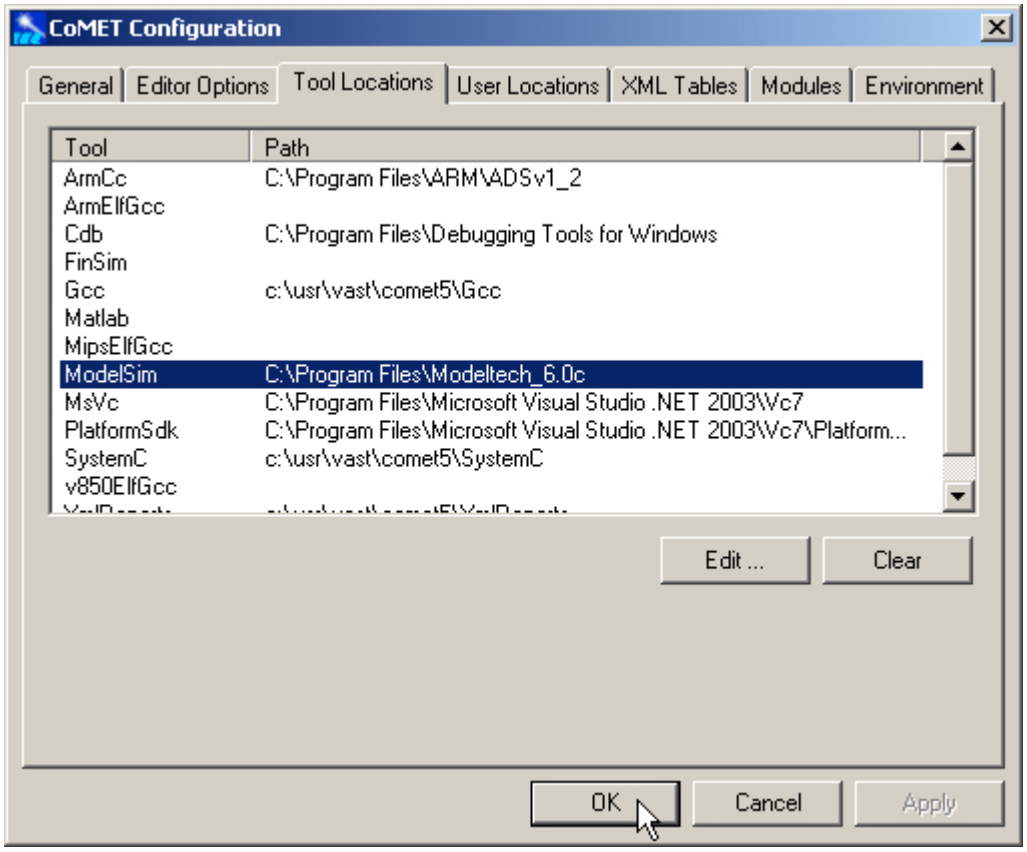
- For details on how to select Fabric port types for mapping with Verilog ports see *Selecting a Fabric Port Type for a Verilog Port*, *page 188*.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly
authorized in a written agreement with VaST.

Running a Verilog Model on CIF

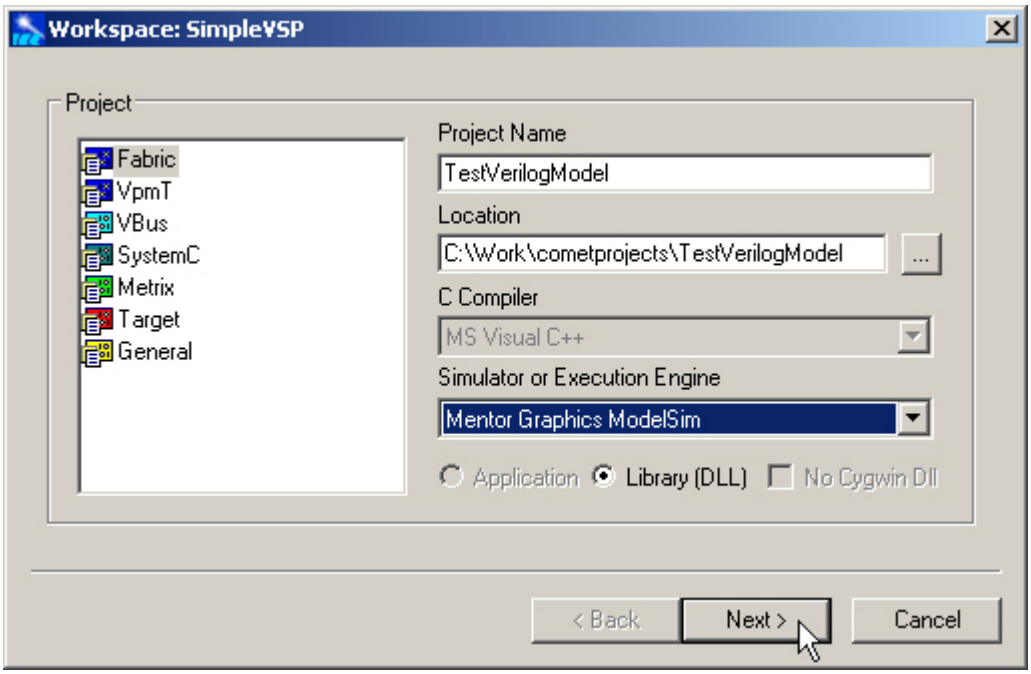
Tool Locations Configuration Settings for Verilog Model

In CoMET Configuration, Tool Locations tab, set ModelSim to point to ModelSim home directory. See *Tool Locations Configuration Settings, page 31*.



Creating a Verilog Model Project

Choose **Workspace/Add New Project**.



Project: Select the Fabric project type.

Project Name: Specify name of the Verilog Project.

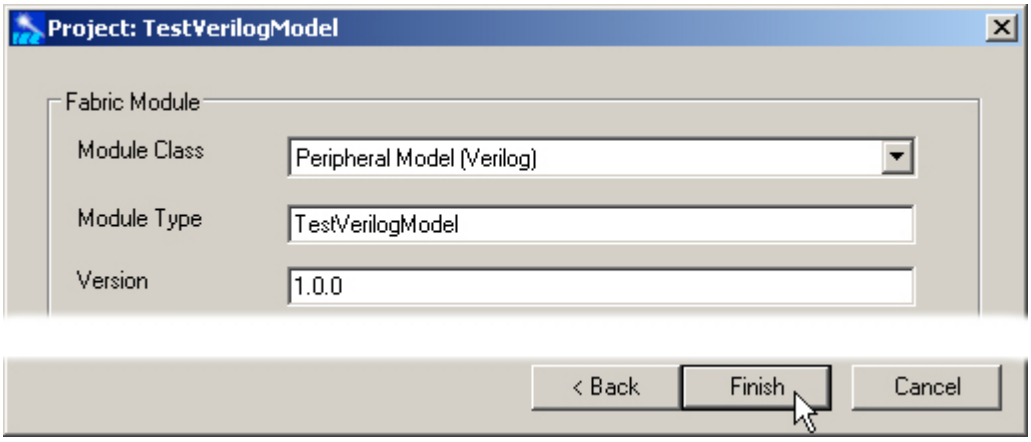
Location: Specify desired location for project files

C Compiler: Select `MS Visual C++`

Simulator or Execution Engine: Select `Mentor Graphics ModelSim`

Click Next.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly
authorized in a written agreement with VaST.



Module Class: Select `Peripheral Model (Verilog)` for a Verilog project.

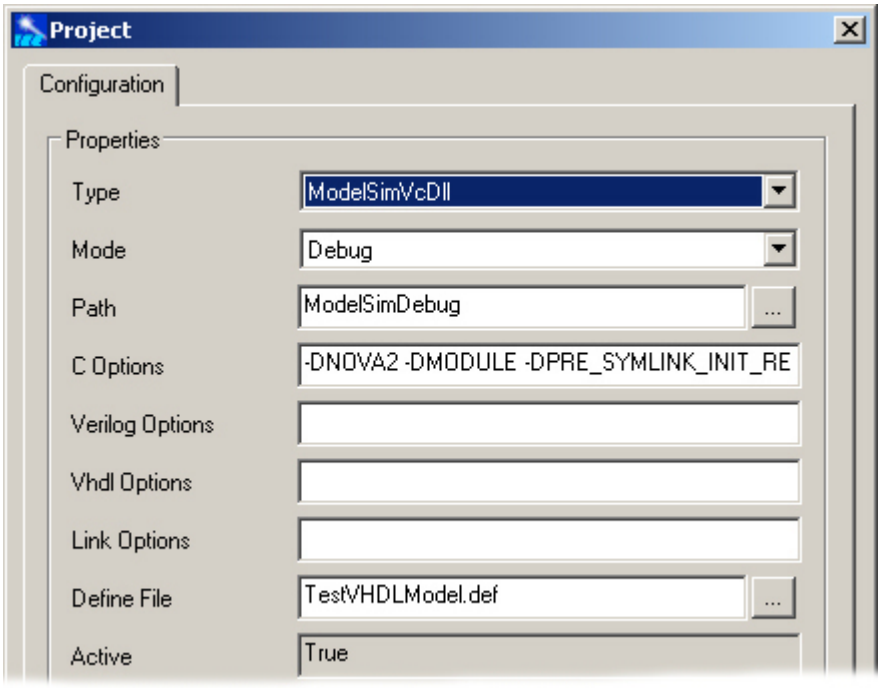
Module Type: Specify the name of the Verilog entity in the Module Type field or accept the default.

Version: Specify the version or accept the default.

Click Finish.

NOTE: The Module Type has to match the name of the Verilog model. The top module can will be a Verilog module, may instantiate other VHDL entities and Verilog modules in the project.

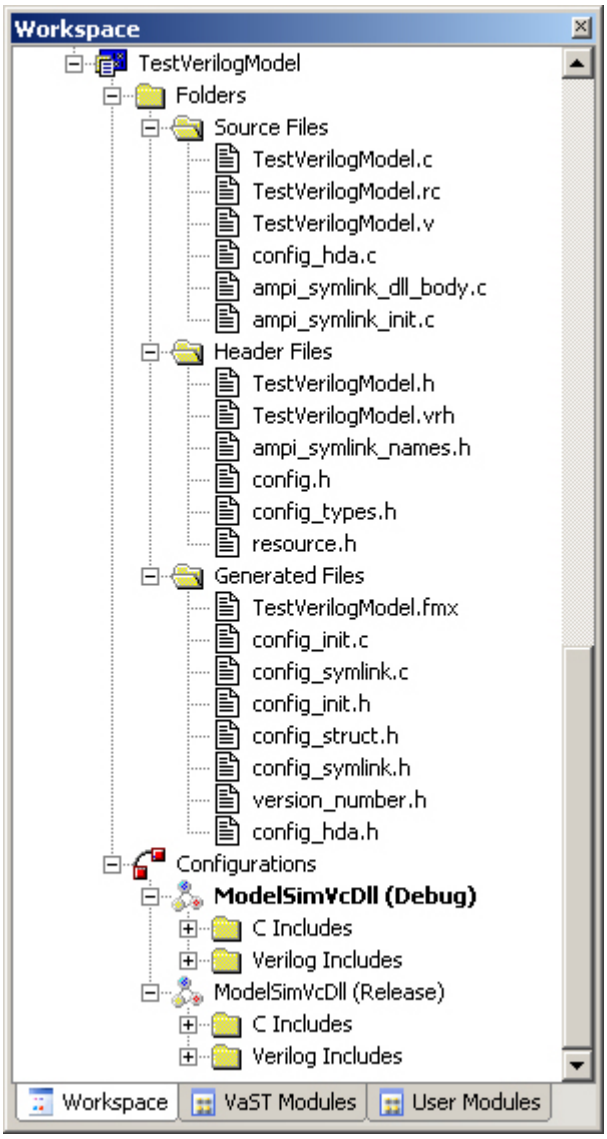
Double-click the Build Configuration to open the Configuration window:



Specify the Verilog complier options for Vlog in the Verilog Options field.

Specify the VDHL complier options for Vcom in the VHDL Options field if there are VHDL entities in the project.

CoMET creates a Verilog Model project from templates.



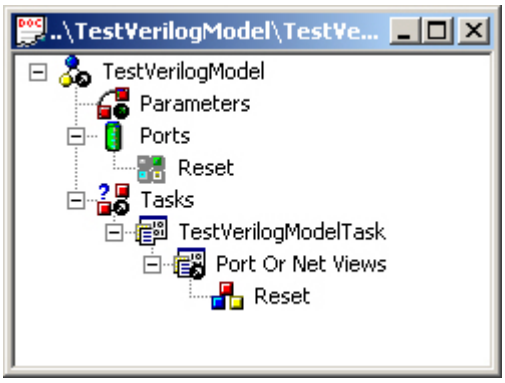
You can now replace the newly created files with the corresponding files from the existing Verilog project:

- Source Files/ProjectName.c – behavioral C for Verilog device
- Source Files/ProjectName.v – Verilog source file
- Header Files/ProjectName.h – behavioral C header
- Header Files/ProjectName.vrh – Verilog header file

Alter the above source files as required. You can also add other VHDL or Verilog source files to the project.

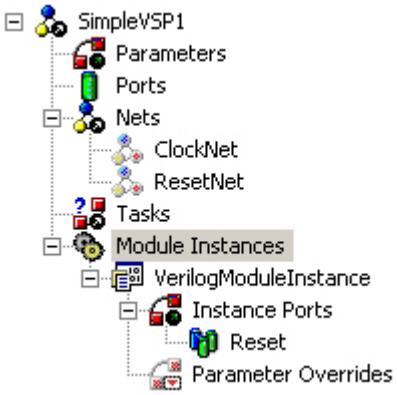
You can now build the Verilog Model.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.



Xml tree view of automatically generated Verilog .fmx file

When the Verilog Model is successfully complied, a module instance of the model can be added to a CIF platform.

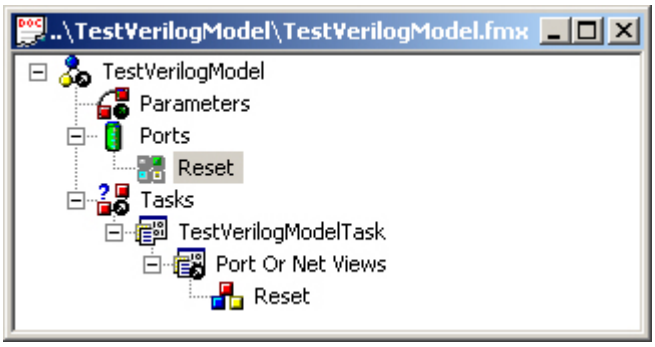


Selecting a Fabric Port Type for a Verilog Port

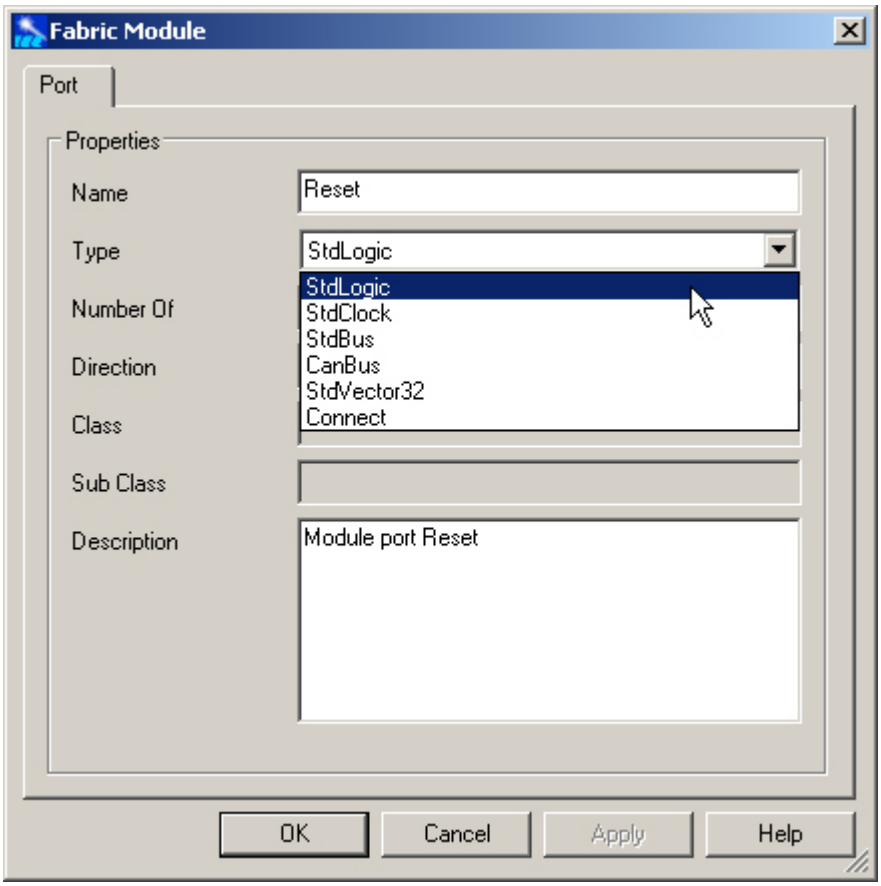
Note: This can only be done after the Verilog source files have been successfully complied and the .fmx file has been successfully updated with the list of ports in the Verilog entity.

To select a particular Fabric Port Type for a Verilog Port:

- Double-click on the **Generated Files/ProjectName.fmx** file. This will open the generated .fmx file document window.



- Double click on the port for which you want to set the Fabric Port Type.



VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

- Select the Fabric Port Type from the **Type** pull down menu. Refer to *Features and requirements, page 180*, for a list of valid Fabric Port types which can be selected for particular Verilog port types.
- Click OK to accept the selection.

Setting the Runtime Options

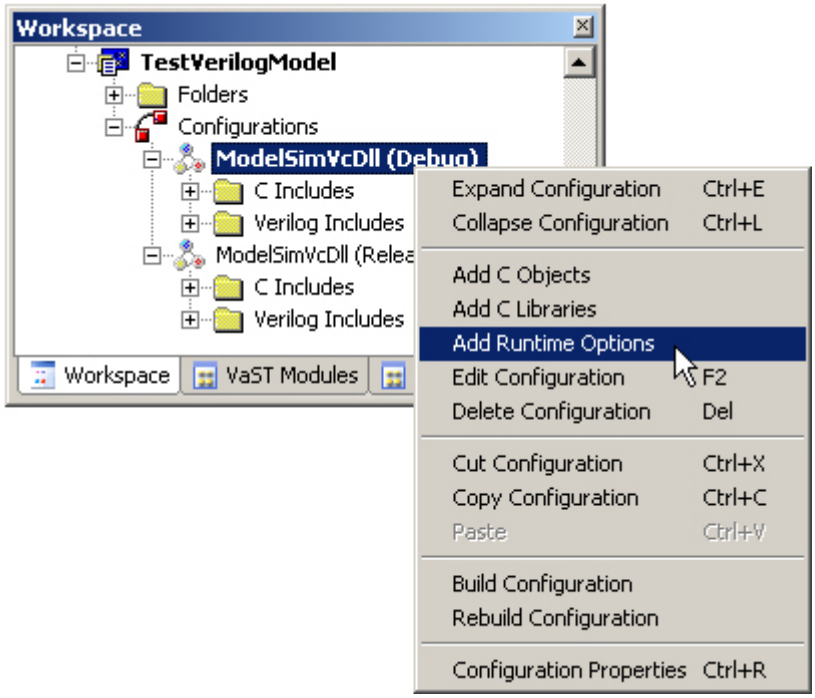
The build configuration of the top module supports a set of runtime options. Runtime Option properties vary for different build configuration types (See *Build Configuration Types, page 110*).

Adding Runtime Option Properties

To gain access to view and change Runtime Options you need to create a Runtime Options node in the Workspace window, under the Configuration node. CoMET creates a Runtime Options node automatically only for top level Fabric projects and VBus projects. You can manually add a Runtime Options node.

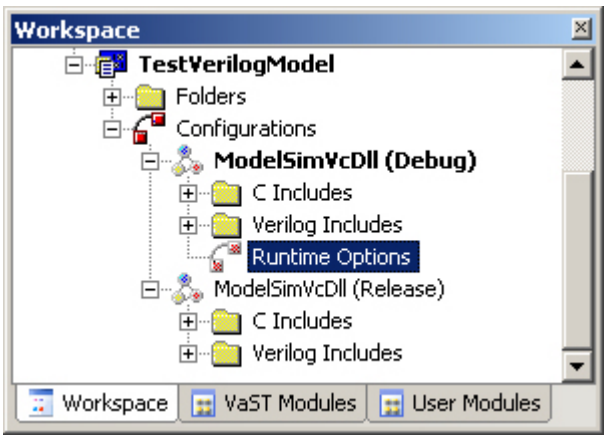
To add runtime options:

- Right click on the appropriate build Configuration node
- Choose **Add Runtime Options** from the context menu.



CoMET creates a new Runtime Options node under the build Configuration node.

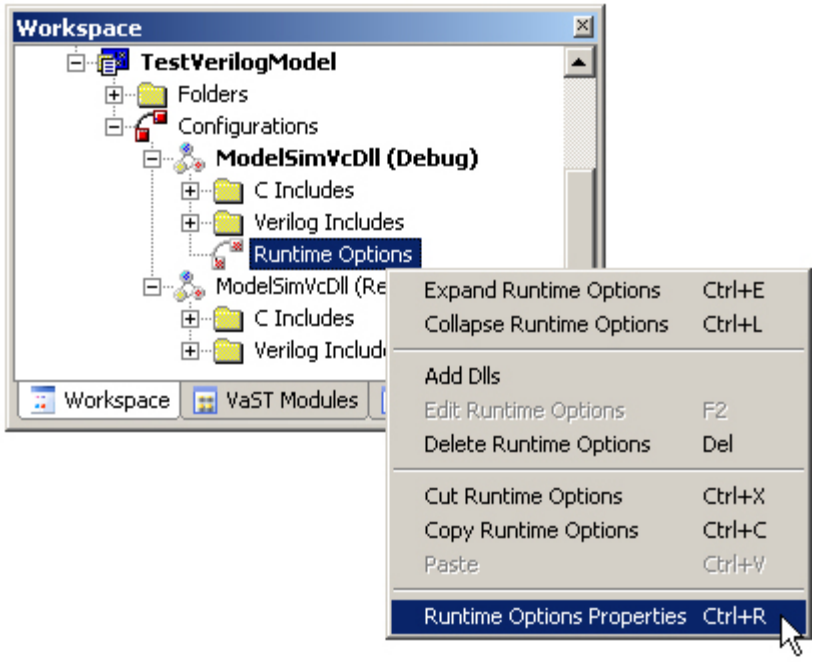
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.



Viewing and Setting Runtime Options Properties

To set the runtime options for the top module:

- Right mouse-click on Runtime Options
- Select **Runtime Options Properties**

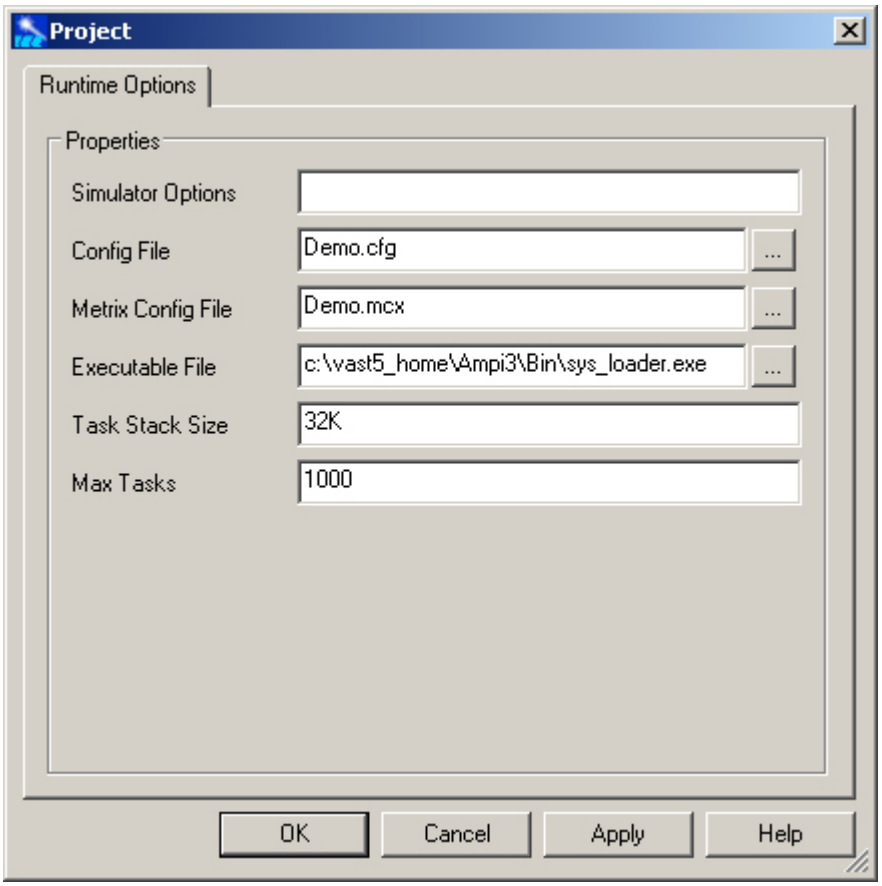


or

- Double-click on **Runtime Options**

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Verilog Project Runtime Options



Simulator Options: These options are passed to the hardware simulator. For the ModelSim Co-Simulation Support there are some additional options you can specify.

1. `-script <commandfile>` : used to specify the name of the file containing the ModelSim simulation commands to be executed.
2. `-vcddump <vcdfilename>` : used to activate the VCD dump of all the signals in the HDL hierarchy to the specified file.
3. `-deltamode` : any HDL port signal change will only be propagated to the CIF at the next delta. If this option is not specified all the signal changes will only be propagated to CIF at the end of the delta cycles.
4. `-sharemem <size>` : Amount of share memory, in bytes, to be allocated for sharing information (for example Port states) between the Nova-2 scheduler and the ModelSim simulator. A default value of 512 K of share memory is allocated.

Config File: The Architecture Configuration File for this project.

Metrix Config File: The path of the associated Metrix Configuration (.mcx) file. See the *Metrix Configuration File User Guide.*

Executable File: This specifies the file that calls the Fabric .dll.

Task Stack Size: Default 32K. May be increased for large projects.

Max Tasks: Default 1000. May be increased for large projects.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

Glossary

AMPI Architecture Modeling Programming Interface.

architecture A term applied to both the process and outcome of designing and creating a specific technological product, including its overall structure, behavior, logical components, and logical interrelationships. An architecture may contain instances of other architectures.

By comparison, the term design has less scope than architecture. An architecture is a design, but most designs are not architectures. A single sub-system or a new function has a design that has to fit within the overall architecture.

A similar term, framework, can be thought of as the structural part of an architecture.

CIF VaST **C**ommunications and **I**nfrastructure **F**abric: an innovative methodology that allows you to design and specify fully executable Virtual System Prototypes (virtual models of a hardware system) that have superior performance and accuracy.

Bus Master A Model that can generate transactions to the StdBus, but cannot receive transactions via the StdBus. A Virtual Processor Model (VPM) is an example of a Bus Master.

Bus Slave A Model that can receive transactions via the StdBus, but cannot generate transactions to the StdBus.

Bus Master/Slave A Model that is capable both of receiving from and generating transactions to the StdBus Protocol Engine.

device	A hardware unit that is capable of providing input to a system or of receiving output from the system, or both.
device model	A virtual representation of a hardware unit that is capable of providing input to a system or of receiving output from the system, or both. Device models (for example VPMs, timers, controllers, and memory) are connected to the bus. Any device model that is driven by target code must be connected to the bus. A device model that does not connect to the bus, such as a console or a test bench, is called an auxiliary device.
Fabric Module	A VaST Communications and Infrastructure Fabric Module. See module.
hardware constant	A fundamental entity of a module. The value of a hardware constant can not change.
hardware value	The value of a hardware variable or hardware constant.
hardware variable	A fundamental entity of a module, such as a register, wire or integer. The value of a hardware variable can change. Its value is the value last assigned to it.
instantiation	The creation of an instance of an object (module).
latency	In a network, a synonym for delay, i.e. an expression of how much time it takes for a packet of data to get from one designated point to another. In a computer system, latency is often used to mean any delay or waiting that increases real or perceived response time beyond the response time desired. Specific contributors to computer latency include mismatches in data speed between the microprocessor and input/output devices and inadequate data buffers.
model	A self-contained module, such as a platform or peripheral device. See module.
module	The definition of a hardware element within a Virtual System Prototype such as a Virtual Platform, a VPM, a Virtual Bus, or a peripheral device. A module describes hardware behavior, functionality, interconnection and hierarchical structure. Modules represent hardware elements and interconnections, and can be instantiated within other modules. Each module generally (but not necessarily) describes a single hardware subsystem.
peripheral device	(See device) Any device that is not part of the essential system (the processor, memory, and data paths) but is situated relatively close by.
ports	The connection points of a module. All modules (including Virtual Platforms, VPMs, peripheral devices, and Virtual Buses) communicate using ports.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

project	A set of files from which you build a Virtual System Prototype. A project may include source files, header files, pre-built modules, DLL's, include files, libraries, and target software files.
register	One of a small set of data holding places that are part of a system and that provide a place for storing variable data. Registers can have arbitrary names.
Software Development Environment	(SDE) A graphical user interface in which you can build interactive real-time embedded software for target system architectures.
StdBus	The VaST Standard Bus (StdBus) Protocol Engine is the main component of a generic Bus Model that can be used for Device and Module Interconnect. The StdBus does not represent a specific proprietary Bus Protocol. It provides a generic StdBus protocol, and emulation of proprietary protocols including AHB, AHB Lite and PCI.
System Engineering Environment	(SEE) The graphical user interface in which you design, build and test Virtual System Prototypes. The System Engineering Environment consists of a workspace, an output window, a source file editor, and tools for building, configuring, executing (or simulating) and debugging Virtual System Prototypes.
task function	A function written using the C language that invokes AMPI functions in order to interact with module elements and interconnections.
Virtual Hardware Platform	See Virtual Platform.
Virtual Platform	(VP) The building block of a Virtual System Prototype. A Virtual Platform may consist of one or more virtual processors (that executes software embedded within the product), peripheral devices, and interconnections, such as buses. Virtual Platforms can also contain other Virtual Platforms in a hierarchical structure. Virtual Platforms may be interconnected.
Virtual Processor Model	(VPM) A device model that simulates the functionality of a hardware processor (CPU).
Virtual System Prototype	(VSP) The virtual representation (or executable specification) of a hardware product that incorporates subsystem design together with the environment that drives the execution (or simulation) of the virtual product. A Virtual System Prototype consists of one or more Virtual Platforms, and (optionally) a test bench and performance and analysis tools.
workspace	The design area (within the CoMET SEE) where you create Virtual System Prototypes.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly
authorized in a written agreement with VaST.

Index

A

adding source files 55

B

Batch mode 147
  CoMET 4 batch files 149, 151
breakpoints
  clearing 142
  continuing execution 143
  managing 143
  setting 142
build output window 138
building the project 137

C

CoMET
  support services 42
  system engineering environment 12
  System Engineering Toolset 5
  web page 42
CoMET Components 5
CoMET configuration 26
CoMET editor 129
CoMET environment 26
configuration
  CoMET 26
  editor options 29
  general configuration 26
Connection View
  XML Table Views 99
Context Menu 14
creating new source files 130

D

debugging
  clearing breakpoints in source code 142
  continuing execution 143
  debugging source code 139
  interrupting execution flow 142
  keys 157
  managing breakpoints 143
  setting breakpoints in source code 142
  software window 141
Disclaimer ii
Document Window 14
  keys 156

E

Editing
  Keys 136, 156
editor
  options 29
  source files 129
environment variables 6
  changing 7
  mandatory 6
Exporting
  Target Binaries 58
  VSP 57
  Workspace 57

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly
authorized in a written agreement with VaST.

F

Fabric
  Update 72
Fabric Module Definition
  opening 63
  Report 100
Fabric Module type 47
files
  source files 129
fmx 14, 27, 63, 97
  Connection View 99
  Report 100
  Update 72

H

hardware requirements 5, 6
help 42
highlighting keywords 131
Hot Keys ..... *See Keys*

I

installing CoMET
  Windows XP 6
interrupting execution flow 142

K

Keys
  Building Projects 157
  Debugging 157
  Document Window 156
  Editing 156
  General 155
  Hot keys 155
  Simulation 157
  Windows 156
  Workspace Window 156
keywords
  highlighting 131

M

mandatory environment variables 6
Matlab Generator 159
menu bar 12
METeor
  Exporting VSP to METeor 57
Module type
  Fabric module 47
Modules Window
  adding a new module library 38
  creating module instances by drag and drop 87
  creating module instances using copy and paste 88
  description 14

O

opening an existing project 50
Output Windows 13

P

Parameter overrides 89
  Viewing in Prototype Report 105
Parameter View
  XML Table View 98
pcx 27
  Parameter View 98
  Report 100
  Update 122
Peripheral Module SystemC
  Module type 48
project
  add existing 50
  adding 45
  adding source files 55
  building 137
  configuring 54
  creating 114
  debugging 141
  opening an existing project 50
  removing 51
  set active project 53
  simulating 141
  viewing properties 52
Prototype
  Viewing structure in Prototype Report ... 105
Prototype Configuration
  Report 100
Prototype Reports 100

R

Report
  .fmx, .pcx 100
  Prototype 100

S

settings
  project 54
Simulating
  keys 157
simulation 141
software window 141
source code
  clearing breakpoints in 142
  debugging 139
  editing keys 136
  managing breakpoints 143
  setting breakpoints in 142

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly authorized in a written agreement with VaST.

source files
adding....55
creating new source files....130
editing....129
Status bar....13
support services....42
SystemC....165
Peripheral Module SystemC....48

T

Target Binaries, Export....58
Target Window....13
tool bar....13
Tools
Matlab Generator....159

U

Update
fmx....72
Update Fabric....72
Update pcx....122
Update Platform....122

V

Validate
Validating VPM behavior....145
Value Change Dump....146
VCD....146
virtual system prototype
model....1
VSP
Exporting....57

W

web page....42
Window
Build output....138
Document....14
Modules....14
Output....13
Target....13
Workspace....13
Windows XP
installing....6
Workspace
creating....43
Exporting....57
Window....13
Window keys....156

X

XML Table Views
Connection View....99
Parameter View....98

VaST SYSTEMS TECHNOLOGY Confidential and Proprietary.
May not be used, disclosed or reproduced except as expressly
authorized in a written agreement with VaST.